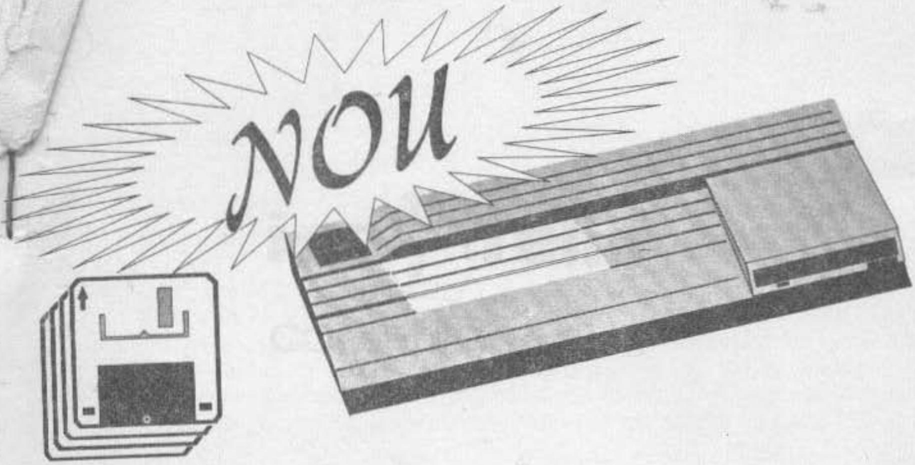


HC 2000

ice felix

COMPUTER S.A.

ice felix

COMPUTER S.A.

HC - 2000

MANUAL TEHNIC

ȘI DE UTILIZARE

CUPRINS

ÎNAINTE DE ORICE VERIFICAŢI CONFIGURAŢIA STANDARD:

- unitatea centrală (tastatura şi microdrive încorporat)
- sursa de alimentare (alimentatorul)
- cablu pentru televizor
- cablu pentru casetofon
- prezentul manual şi manualul de operare CP/M
- discheta de demonstraţii.

Capitolul 1. INTRODUCERE

1.1. Prezentare generală

Stimate cumpărător, acest manual este făcut cu intenţia de a ghida primii paşi în utilizarea calculatoarelor din familia Home Computers - microcalculatoare cu destinaţie educaţională, divertisment, calcule ştiinţifice şi inginereşti.
Un calculator personal este folosit de o singură persoană, spre deosebire de alte tipuri de calculatoare (micro sau mini sisteme) la care pot lucra simultan mai multe persoane.
Calculatoarele personale sunt şi ele de două feluri:

-calculatoare personale profesionale (PERSONAL COMPUTER);
-calculatoare personale familiale (HOME COMPUTER).

Acestea din urmă au un preţ accesibil pentru a putea fi cumpărate pentru acasă. Calculatoarele tip HC fac parte din această grupă.
Manualul se adresează tuturor, fără a cere o pregătire în electronică sau informatică. El nu va arăta cum se construieşte un calculator ci din ce este format, cum se foloseşte şi ce se poate ataşa la el pentru a-i putea îmbunătăţi performanţele.

1.2. Caracteristici tehnice

CPU -Z 80 A - microprocesor pe 8 biţi cu ceas de 3,546 MHz;
ROM - 48 Kocteţi - memorie ROM din care 16 K pentru interpretorul BASIC, 16 K pentru funcţiile BIOS CP/M şi 10 K pentru funcţiile specifice Interfeţei 1 (IF1).
RAM - 64 K - memorie RAM din care 48 K disponibili în mod de lucru BASIC SINCLAIR şi 56 K disponibili în mod de lucru CP/M.
TASTATURA: - extinsă de 50 taste, similare celei de pe maşinile de scris şi care include şi tastele funcţionale BASIC şi CP/M.
DISPLAY: - afişare pe televizor alb/negru sau color PAL pe canalul 8, monitor RGB sau monitor PAL.
- rezoluţie: 192*256 pixeli (24*32 caractere).
- realizează punct, linie, cerc, arc de cerc de înaltă rezoluţie grafică.
- 16 caractere grafice predefinite, 21 de posibilităţi de definire grafică.
- textul scris pe SCREEN are 32 caractere pe 24 linii.

SUNET: - sunetul auzit în difuzorul calculatorului cuprinde circa 10 octave realizate prin comanda BASIC: BEEP.

CULORI: -detaliile în plan apropiat cât şi în plan îndepărtat se realizează prin culoare, strălucire şi flash cu setul de instrucţiuni: INK, PAPER, BORDER, BRIGHT şi FLASH.
- codul culorilor este controlabil de la tastatură.

- comanda INVERSE 1 inversează fundalul cu cerneala, iar OVER 1 realizează supraimprimarea.

INTERFEȚE ÎNCORPORATE:

- interfața casetofon audio, 1500 bauds;
- interfața joystick compatibil Sinclair (IF2);
- interfața de disc flexibil 3,5 țoli, 80 piste. dublă față, 720K (unitatea centrală incluzând și unitatea de disc);
- interfața serială compatibilă CCITT V24 (RS-232C);
- interfața pentru adăugarea unui minidisc exterior de 3,5 țoli sau de 5,25 țoli, dublă față, 40 piste, 360K.

SOFTWARE: - interpretor BASIC 16K inscris în memorie eprom.

- LOGO, FORTH, PASCAL, BETA BASIC și altele pe casetă.
- JOCURI pe casetă sau dischetă.
- orice program care rulează sub sistemul de operare CP/M.

1.3. Instalare

Calculatorul se alimentează prin intermediul alimentatorului de la rețeaua de curent alternativ de 220V.

1 - mufa antenă televizor (canalul 8);
2 - conector monitor RGB sau video complex;
3 - conector joystick-uri standard Sinclair (IF2);
4 - conectorul de extensie BUS. Acesta permite accesul utilizatorului la magistralele microprocesorului Z80 și cuplarea altor interfețe exterioare (de exemplu interfața pentru joystick Kempston și creion optic livrată tot de FELIX COMPUTERS S.A.);
5 - mufa pentru cuplarea casetofonului audio;
6 - mufa alimentare calculator;
7 - mufa interfața serială și rețea;
8 - mufa alimentare minidisc exterior (opțională, numai pentru anumite modele de disc!);
9 - mufa cuplare minidisc exterior (opțional);
10 - buton RESET

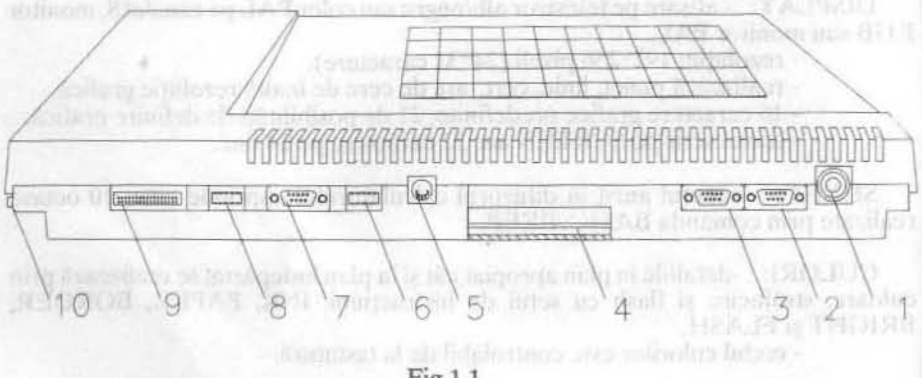


Fig.1.1

Pentru punerea în funcțiune și utilizarea calculatorului urmăriți secvența de mai jos:

1. Introduceți fișa alimentatorului în mufa numărul 6 (vezi desenul anterior) din spatele calculatorului.

2. Introduceți ștecherul alimentatorului într-o priză de curent alternativ 220V/50Hz. Puneți pe ON întrerupătorul de pe carcasa alimentatorului.

Din acest moment HC-ul 2000 funcționează. Dacă apăsați tastele auziți bipuri sonore. Dacă nu le auziți apăsați butonul RESET și încercați din nou. Butonul RESET se află așezat în partea dreaptă a microdrive-ului, ceva mai jos decât aceasta, pentru a nu fi atins din greșeală, în timpul lucrului.

În momentul în care apăsați tastele, calculatorul primește comenzile dumneavoastră. Pentru a putea dialoga aveți nevoie de un dispozitiv de afișare. Cel mai simplu este un televizor alb/negru sau color.

3. Conectați cablul de televizor. Introduceți mufa RCA în locul notat tv (mufa nr. 1), iar celălalt capăt în mufa de antenă a televizorului.

ATENȚIE: Nu recomandăm folosirea televizoarelor pe tuburi.

4. Porniți televizorul și acordați-l pe canalul 8 până ce obțineți o imagine clară.

După instalare, pe ecran, în partea de jos, trebuie să apară un mesaj care reprezintă numele calculatorului și firma constructoare . În caz că nu apare apăsați din nou butonul RESET. Din acest moment calculatorul este pregătit pentru dialog.

Dacă vă aflați pentru prima dată în fața unui astfel de calculator e bine să aflați mai întâi posibilitățile sale, resur- sele hard și soft. Cel mai simplu pentru aceasta este folosirea discului sau casetei de demonstrații.

Vom descrie în continuare procedura de încărcare a primului program de pe caseta de demonstrație (în cazul în care ați primit o dischetă de demonstrație, citiți secțiunea de încărcare a programelor descrisă în capitolul 4, care explică operarea minidiscului).

Pentru aceasta vă este necesar un casetofon. Acesta nu trebuie să fie foarte sofisticat. Este necesar să prezinte o mecanică sigură, fără fluctuații de bandă și un cap cât mai puțin uzat și reglat pe un azimut corespunzător (așa cum îl livrează fabricantul).

5. Introduceți cablul de casetofon în calculator în locul notat cas (mufa nr. 5), iar capătul celălalt în casetofon pe mufa LINE.

6. Introduceți caseta în casetofon și poziționați-o la început.

7. Priviți tastatura și localizați urmatoarele taste: J, P, SS, CR. Apăsați J apoi țineți cu un deget SS și apăsați de două ori tasta P. Pe ecran apare LOAD"" . Apăsați apoi RETURN. Ecranul se face alb.

8. Porniți casetofonul. În acest moment calculatorul va schimba BORDER-ul în albastru și apoi în roșu. Pentru posesorii de televizoare alb/negru, dintr-o culoare deschisă într-una mai închisă și invers, până când începe programul pe casetă. Din acest moment pe BORDER apar dungi colorate. Se încarcă programul. Urmăriți și executați mesajele de pe ecran. După ce v-ați familiarizat cu tastatura și caseta citiți mai departe manualul, dar mai înainte să lămurim niște termeni folosiți anterior:

-ROM: (Read-Only-Memory) este o memorie al cărui conţinut este stabilit prin fabricaţie şi care nu poate fi schimbat ci numai "citit". Veţi constata că de câte ori scoateţi de sub tensiune calculatorul interpretorul BASIC nu dispare din calculator.

-RAM: (Random-Access-Memory) este memoria de lucru curent a calculatorului. În ea se poate "scrie" şi "citi" ceea ce doriţi, ori de câte ori apelaţi calculatorul, atâta timp cât acesta este alimentat. La întreruperea alimentării pierde ce are înscris prin program de dumneavoastră.

-HARDWARE: Specialiştii numesc astfel echipamentele ce alcătuiesc calculatorul în totalitatea lor.

-SOFTWARE: Este tot ce reprezintă programe.

Pentru a scrie ceva avem nevoie de hârtie şi cerneală. Pentru a defini acest lucru notăm partea activă a ecranului (SCREEN) cu PAPER = hârtie, coală; ceea ce scriem notăm INK, iar pentru a separa PAPER-ul de marginile ecranului care ar putea ascunde la colţuri notiţele noastre, folosim BORDER-ul care centrează PAPER- ul în aşa fel încât în orice caracter de pe ecran să fie vizibil.

1.4.Tastatura

Dupa cum desigur aţi observat calculatorul are un număr de 50 de taste. Tastatura seamănă foarte mult cu claviatura unei maşini de scris, însă este mai complicată deoarece fiecare tastă are cel puţin şase semnificaţii. Prin semnificaţii înţelegem litere mici sau mari, cifre, caractere speciale (de exemplu +,-,?,*,$,%,etc.) sau cuvinte cheie (de exemplu INPUT, PRINT, RUN,etc.). Cuvântul cheie este un cuvânt în limba engleza care are o semnificaţie foarte precisă pentru calculator. Pentru exemplificare tasta i are următoarele semnificaţii: i, I, INPUT, AT, CODE, IN.

CODE	AT
I	
IN	INPUT

Pentru a alege semnificaţia dorită de pe o tastă trebuie să cunoaştem "modurile de lucru ale calculatorului".

Calculatorul are cinci moduri de lucru: K, L, C, E si G.

Modul de lucru în care se găseşte calculatorul ne este indicat de o literă mare clipitoare numită "cursor". Cursorul ne indică şi locul de pe ecran unde va apărea următoarea semnificaţie.

MODUL K: modul cuvintelor cheie "keywords"

Dacă suntem în modul K şi apăsăm o cifră, pe ecran apare cifra respectivă, cursorul rămânând în K. Dacă apăsăm o literă, pe ecran apare cuvântul cheie din dreapta jos a tastei (de exemplu INPUT pentru tasta i), cursorul trecând automat în modul L.

Atenţie! Întotdeauna cuvintele cheie vor fi scrise direct apăsând tasta corespunzătoare şi nu literă cu literă.

În modul K se intră în următoarele cazuri:
-la începutul fiecărei linii
-după semnul : (două puncte), care separă instrucţiunile de pe aceeaşi linie.
-după cuvântul cheie THEN.

MODUL L: literele mici şi mari. Apare imediat după modul K sau E. În modul L, dacă se apasă o cifră apare cifra respectivă, iar dacă se apasă o literă apare litera mică respectivă, cursorul rămânând în L. Dacă dorim să scriem litere mari atunci apăsăm simultan CAPS SHIFT (CS) şi tasta respectivă.

MODUL C: numai litere mari, (capitals). Dacă dorim să scriem numai cu litere mari intrăm în modul C apăsând simultan CS şi 2 sau direct tasta C.LOCK.

Din C se iese apăsând iarăşi tasta C.LOCK.

OBSERVAŢIE: Dacă dorim să scriem semnificaţia din dreapta sus trebuie să fim în unul din modurile K, L sau C şi să apăsăm simultan SYMBOL SHIFT (SS) şi tasta respectivă.

MODUL E: extins. Se utilizează pentru a scrie semnificaţiile din stânga sus şi jos ale tastelor. În modul E se intră apăsând simultan CS şi SS sau tasta ESEND MODE. Pentru a scrie semnificaţia din stânga sus intrăm în modul E, după care apăsăm tasta corespun-zătoare.

Pentru a scrie semnificaţia din stânga jos intrăm în modul E, după care apăsăm simultan SS şi tasta corespunzătoare. După prima apăsare pe tastă cursorul trece din modul E în modul L.

MODUL G: grafic "graphics". Apare după ce se apasă simultan CS şi 9 sau tasta GRAPH şi ţine până când se apasă 9 sau iarăşi GRAPH. În modul G se pot scrie simboluri grafice "mozaic", folosind tastele 1-8 cu şi fără CS. Tasta 0 se utilizează pentru a şterge caracterul din stânga cursorului. Tot în modul G putem să definim propriile noastre caractere grafice.

Dacă o tastă este apăsată mai mult de 0,7 secunde, ea va fi scrisă în mod repetat atâta timp cât o apăsăm.

Ceea ce scriem la tastatură va apare în partea de jos a ecranului în timp ce se tastează, fiecare caracter fiind inserat chiar înaintea cursorului. Cursorul poate fi deplasat spre stânga cu CS şi 5, şi spre dreapta cu CS şi 8, fără a şterge caracterele respective. De asemenea se pot utiliza săgeţile plasate în stânga şi dreapta tastei BLANK. Caracterul dinaintea cursorului poate fi şters indiferent de modul în care ne găsim, cu DELETE (CS şi 0).

OBSERVAŢIE Tot ce am scris în partea de jos a ecranului poate fi şters apăsând EDIT (CS şi 1) urmat de RETURN.

Modul E: se apasă tasta	CODE	IN	Modul K, L și C: se apasă SS și tasta
Modul L și C	I		
Modul E: se apaă SS și tasta	IN	INPUT	Modul K: se apasă tasta

1.5. Limbaje de programare

Un calculator poate să facă practic orice. Important este ca noi să știm să-i spunem ceea ce trebuie să facă. Acest lucru se face prin realizarea unui "program". Programul reprezintă o înșiruire de instrucțiuni așezate într-o ordine foarte precisă, prin intermediul căruia dirijăm calculatorul pas cu pas în ceea ce trebuie să facă. Bineînțeles că cel puțin deocamdată calculatorul nu înțelege "limbajul natural", limbajul în care comunicăm noi oamenii, de aceea fiind necesar să învățăm "limba" pe care o știe el și care se cheamă "limbaj de programare".

Deoarece la ora actuală în lume există mii de tipuri de calculatoare, cred că înțelegeți necesitatea existenței celor peste două sute de limbaje de programare. De ce așa de multe?. Nu era suficient un singur limbaj de programare?.

Răspunsul constă în faptul că de obicei un limbaj de programare acoperă, cu eficiență maximă, doar un domeniu, fiind mai puțin eficient în celelalte. De exemplu limbajul FORTRAN (FORmula TRANslator) este cel mai potrivit limbaj pentru rezolvarea problemelor tehnico-științifice. Pentru probleme de gestiune, deci economico-financiare, cel mai cunoscut este limbajul COBOL (COmmon Business Oriented Language).

Calculatoarele HC, care sunt așa cum am arătat, calculatoare pentru acasă, deci pentru publicul larg, folosesc un limbaj accesibil tuturor numit BASIC (Beginners All-purpose Symbolic Instruction Code). Aceasta în românește s-ar putea traduce: limbaj de programare pentru începători. După cum îi spune și numele, acest limbaj de programare poate fi învățat de toți cei care doresc să pătrundă în universul fascinant al calculatoarelor. Dacă vreți să utilizați calculatorul ca "beneficiar" cu programe "gata făcute" nu mai aveți practic multe de învățat. Un scurt ghid de BASIC este suficient. De obicei însă în practică doriți să aveți unele facilități cu ajutorul calculatorului d-vs și pentru aceasta este necesar să cunoașteți cât mai multe. În felul acesta puteți să faceți singuri aceste programe. Pentru programe simple consultați manualul BASIC pentru HC sau altele care folosesc același limbaj de programare (ZX Spectrum, CIP, Spectim, etc.). Pentru programe mai evoluate folosiți limbajul cod mașină.

1.6. Câte ceva despre HC-2000

Așa cum am arătat mai sus calculatoarele HC-2000 lucrează în limbaj BASIC. Acest interpretor este compatibil Sinclair. Din acest punct de vedere toate tipurile de HC-uri sunt perfect compatibile BASIC. Singurele diferențe sunt din punct de vedere hard.

HC-2000 are avantajul că poate lucra și în CP/M, care este un sistem de operare profesional. El vă oferă o altă cale de a lucra cu minidiscul și imprimanta, mult mai multe coloane de caractere afișate pe ecran, dar cel mai important lucru este faptul că vă permite să rulați pe calculatorul dvs. programe "serioase" care rulează de regulă pe calculatoare mai mari: M-118, CUB-Z sau orice alt calculator dotat cu microprocesor INTEL 8080 sau ZILOG Z80 și pe care este instalat CP/M-ul. Acum veți putea lucra de exemplu cu programe ca: WordStar, Dbase II, Turbo Pascal, "C" Aztec, M80, L80, LIB80, Cod80, ZSID, POWER, DIP și multe altele!

Capitolul 2. ELEMENTE DE PROGRAMARE ŞI EDITARE

2.1 Utilizarea tastaturii

Am arătat în capitolul precedent că tastatura HC-ului este similară unei maşini de scris. Am mai arătat că o tastă are până la şase semnificaţii. Cum se tastează fiecare funcţie v-aţi familiarizat deja în urma lecturării primului capitol şi vizionării casetei sau dischetei de demonstraţii. Mai rămâne de menţionat că la înscrierea simbolurilor pe tastatură au fost folosite următoarele prescurtări:

RAND în loc de RANDOMIZE
BRGT în loc de BRIGHT
INV în loc de INVERSE
CS în loc de CAP SHIFT
SS în loc de SIMBOL SHIFT
SCR$ în loc de SCREEN$
CONT în loc de CONTINUE

2.2 Modul de afişare

Ecranul de afişare are 24 de linii, fiecare cu 32 de caractere.
Ecranul are două părţi. Partea de sus de 22 de linii este folosită pentru listarea instrucţiunilor sau a rezultatelor programului. Când această parte este plină, calculatorul face "scroll". Pentru a vedea toate liniile, calculatorul se opreşte şi apare mesajul "scroll".

Apăsarea tastelor N, SPACE sau STOP va întrerupe programul şi va afişa mesajul:
D BREAK - CONT repeats

Orice altă tastă determină calculatorul să facă scroll. Partea de jos a ecranului este folosită pentru comenzi de intrare, linii de program, tipărirea datelor de intrare cât şi pentru mesaje.

2.3 Programe, linii de program şi editarea programelor utilizând EDIT şi săgeţile, RUN, PRINT STOP, IN, INPUT, DATA, BREAK

Limbajul BASIC admite două tipuri de instrucţiuni: numerotate şi nenumerotate. Instrucţiunile nenumerotate sunt executate imediat după apăsarea tastei RETURN. Instrucţiunile numerotate sunt stocate ca linii de program. Numerele de linie trebuie să fie întregi, între 1 şi 9999. Listarea şi execuţia unui program se fac prin ordonarea programului după numărul de linie. De aceea este indicat ca la scrierea unui program să se păstreze spaţii între numerele a două linii consecutive, dând astfel posibilitatea înserării cu uşurinţă de linii noi. O linie de

program poate conţine una sau mai multe instrucţiuni. Separarea instrucţiunilor dintr-o linie se face cu caracterul ":".

Cursorul indică linia curentă asupra căreia se pot face modificări sau după care se pot insera alte linii. De obicei cursorul se află pe ultima linie introdusă, dar poziţia lui poate fi deplasată în sus sau în jos prin apăsarea simultană a tastei CAPS SHIFT şi a săgeţilor.

În continuare vor fi prezentate exemple de programe în care sunt trecute în revistă câteva instrucţiuni BASIC, punându-se accentul pe facilităţile de editare ale sistemului.

Exemplul 1. Să se tipărească suma a două numere.
După ce se vor introduce liniile (în ordinea menţionată):

```
20 PRINT a
10 LET a=10
```

se constată că programul se afişează pe ecran în permanenţă ordonat după numărul de linie. Până acum s-a introdus primul număr. Pentru a-l introduce pe al doilea, se scrie linia:

```
15 LET b=15
```

Pentru tipărirea sumei este necesar ca linia 20 să aibă forma:

```
20 PRINT a+b
```

S-ar putea rescrie linia, dar este mai uşor să se facă uz de facilitatea EDIT. Pentru aceasta se coboară cursorul de la linia 15 la linia 20, acţionând tasta ↓. În continuare se acţionează tasta EDIT; în partea de jos a ecranului va apare o copie a liniei curente (în exemplul prezentat, linia 20). Se acţionează tasta → până când cursorul L se deplasează la sfârşitul liniei şi apoi se introduc +b fără RETURN.
Ultima linie a ecranului va arăta acum astfel:

```
20 PRINT a+b
```

Cu RETURN, vechea linie 20 va fi înlocuită cu cea nouă. Se execută acest program utilizând RUN şi RETURN; ca urmare pe ecran va apare afişat rezultatul operaţiei a + b. Apăsând RUN şi RETURN, programul este executat identic. După terminarea execuţiei programului, în memorie rămâne înregistrată ultima valoare a fiecărei variabile din program. Ele pot fi vizualizate printr-o instrucţiune PRINT neetichetată. Această operaţie este necesară la depanarea programului. Pentru a şterge ultima linie a ecranului se utilizează EDIT. Se introduce o succesiune de caractere (fără RETURN) care vor fi şterse folosind una din metodele:
1. acţionarea tastei DELETE până când linia este ştearsă în întregime;
2. acţionarea tastei EDIT; pe ultima linie a ecranului apare o copie a liniei curente. Cu RETURN acum, linia curentă rămâne nemodificată, iar ultima linie a ecranului este ştearsă.
Presupunem că se introduce din greşeală linia:

12 LET b=8

Ea va putea fi ştearsă scriind:

12 (cu RETURN desigur)

Se observă că a dispărut cursorul programului. Dacă se acţionează ↑, cursorul va apare pe linia 10, în timp ce dacă se acţionează ↓, va apare la linia 15. Se scrie:

12 (şi RETURN)

Din nou cursorul programului va fi ascuns între liniile 10 şi 15.

Acţionând acum EDIT, linia 15 va apare în zona de editare. Cînd cursorul programului este ascuns între două linii, EDIT aduce în josul ecranului linia care are numărul de linie imediat următor. Se scrie acum:

30 (şi RETURN)

De această dată cursorul este ascuns după sfârşitul programului.
Cu comanda:

LIST 15

pe ecran se obţine:

15 LET b=15
20 PRINT a+b

Instrucţiunea LIST 15 produce listarea începând cu linia 15 şi pune cursorul programului la linia 15. Pentru un program foarte lung, LIST va fi o metodă mai utilizată de mutare a cursorului decât săgeţile. Aceasta ilustrează o altă utilitate a numerelor de linie; ele acţionează ca nume ale liniilor de program astfel încât se pot face referiri la ele în acelaşi mod în care se fac referiri la numele de variabile. LIST (neurmat de un număr) determină listarea de la începutul programului.

O altă comandă este NEW. Efectul ei constă în ştergerea programelor şi variabilelor din memoria calculatorului.

EXEMPLUL 2. Să se scrie un program care transformă temperatura din grade Fahrenheit în grade Celsius.

```
10 REM conversia temperaturii
20 PRINT "grade F", "grade C"
30 PRINT
40 INPUT "introduceţi gradele F.",f
50 PRINT f,(f-32)*5/9
60 GO TO 40
```


Este necesar să fie introdusă pe rând fiecare literă pentru a obţine "conversia temperaturii" în linia 10. În linia 60 se obţine GO TO acţionând tasta G (deşi conţine spaţiu, GO TO constituie un singur cuvânt cheie).

Rulând programul, se va vedea pe ecran capul de tabel tipărit de linia 20. Linia 10 este ignorată de calculator, instrucţiunea REM introducând un comentariu în textul sursă. Comanda INPUT din linia 40 aşteaptă să fie introdusă o valoare pentru variabila F; se introduce un număr şi se acţionează apoi RETURN. Calculatorul afişează rezultatul şi nu se opreşte din rulare, ci aşteaptă alt număr (datorită saltului din linia 60). Programul se poate opri prin acţionarea tastei STOP în momentul în care pe ecran apare scris:

introduceţi gradele F.

Calculatorul întoarce mesajul:

H STOP in INPUT 40:1

care precizează de ce şi unde s-a oprit din rulare (în prima instrucţiune din linia 40). Pentru a continua programul se introduce CONTINUE şi calculatorul va aştepta alt număr. CONTINUE determină rularea programului de la linia de la care se oprise execuţia (linia 40). Se scrie linia 60 sub forma:

60 GO TO 31

În execuţie, această variantă se comportă identic cu varianta precedentă. Dacă numărul liniei într-o comandă GO TO se referă la o linie inexistentă, atunci se sare la linia imediat următoare numărului dat. Acest lucru este valabil şi pentru comanda RUN (de fapt RUN are acelaşi efect cu RUN 0). Dacă tipărim numere până când se umple ecranul, calculatorul va muta întreaga parte de sus a ecranului cu o linie pentru a face loc, pierzând astfel capul de tabel. Când am terminat de tipărit, programul se poate opri cu STOP urmat de RETURN. Lista de instrucţiuni a programului se poate afişa după întrerupere apăsând RETURN. Se analizează instrucţiunea PRINT din linia 50. Virgula utilizată aici determină începerea tipăririi fie în marginea din stânga, fie în mijlocul ecranului, în funcţie de ce urmează după virgulă. În acest caz tipărirea temperaturii în grade Celsius are loc în mijlocul liniei.

Caracterul punct şi virgulă ";" determină tipărirea şirului următor imediat după şirul precedent. Se poate vedea aceasta dacă în linia 50 se înlocuieşte caracterul "," cu ";". Alt semn de punctuaţie ce poate fi utilizat în comenzi PRINT este apostroful "'". El determină saltul cursorului la începutul liniei următoare şi continuarea tipăririi din acel punct, ca şi cum elementele despărţite prin "'" ar fi fost sub incidenţa unor comenzi PRINT succesive. Pentru ca instrucţiunea PRINT să nu determine saltul la linia următoare, este necesar ca PRINT-ul precedent să se termine cu "," sau cu ";". Pentru exemplificare, să se substituie linia 50 pe rând cu liniile:

```
50 PRINT f,
50 PRINT f;
50 PRINT f
50 PRINT f'
```

Se constată că varianta cu "," împarte totul în două coloane, cea cu ";" scrie totul compact, cea fără semn de punctuaţie şi cea cu "'" scriu un număr pe o linie. În memorie pot exista simultan mai multe programe cu condiţia ca numerele de linie să fie în intervale disjuncte.

EXEMPLUL 3

```
100 INPUT n$
110 PRINT "Salut ",n$," !"
120 GO TO 100
```

Acesta este un program care poate coexista în memorie cu programul din exemplul 2 întrucât unul are numerele de linie în intervalul 0...60, iar celălalt în 100...120. Pentru lansarea în execuţie a programului din exemplul 3 se dă comanda RUN 100. Execuţia unei comenzi RUN determină ştergerea ecranului şi a tuturor variabilelor, după aceasta executând şirul instrucţiunilor programului. Dacă nu se doreşte iniţializarea variabilelor şi ştergerea ecranului, se poate utiliza comanda GO TO 100.

La execuţia programului din exemplul 3 se observă că pe ecran apare "L" care indică faptul că se doreşte citirea unui şir de caractere. Sistemul admite ca o instrucţiune INPUT să se comporte similar cu o instrucţiune de atribuire, dar numai pentru cazul citirii de variabile de tip şir de caractere. Pentru aceasta se şterg ghilimelele (utilizând ← şi DELETE) şi se introduce numele unei variabile de acelaşi tip. Introducerea unui nume de variabilă determină căutarea valorii acelei variabile ce trebuia citită de la tastatură.

De exemplu, dacă la execuţia programului din exemplul 3 la prima solicitare de şir de caractere se introduce "ANA", valoarea variabilei n$ va deveni n$ = "ANA", la următoarea citire se introduce "MARIA", n$ devine n$ = "MARIA". La execuţia următoarei instrucţiuni INPUT se va introduce n$; în acest caz se caută valoarea vechii variabile n$ şi i se asociază variabilei n$.

Deci comanda se comportă similar cu LET n$ = n$; în urma acestei instrucţiuni va fi n$ = "MARIA", deci instrucţiunea PRINT din linia 110 va tipări:

Salut MARIA !

Uneori, din greşeală, se scrie un program ce rulează la infinit, cum este următorul:

```
200 GO TO 200
RUN 200
```

Pentru oprirea execuţiei se acţionează BREAK (CAPS SHIFT şi SPACE) şi calculatorul răspunde cu mesajul:

L BREAK into program, 200:1

La sfârşitul fiecărei instrucţiuni programul verifică dacă aceste taste sunt acţionate; dacă da, rularea este oprită. Tasta BREAK poate fi utilizată de asemenea

când sunt conectate casetofonul sau imprimanta, în cazul în care calculatorul aşteaptă ca aceste periferice să efectueze o comandă. Mesajul produs în acest caz este diferit:

D BREAK - CONT repeats

Comanda CONTINUE în cazul lucrului cu casetofonul sau imprimanta repetă instrucţiunea unde programul a fost oprit.

Listingurile automate sunt acelea care nu rezultă în urma unei comenzi LIST, ci au loc după introducerea unei linii noi. De reţinut este faptul că linia curentă apare întotdeauna pe ecran şi în mod normal în poziţia centrală. Calculatorul memorează numărul liniei curente şi, de asemenea, al primei linii din partea de sus a ecranului. Când încearcă să listeze, primul lucru pe care-l face este să compare prima linie de pe ecran cu linia curentă. Dacă prima linie de pe ecran este mai mare ca număr decât linia curentă, atunci cursorul va apare pe prima linie a ecranului. Astfel listarea constă în tipărirea pe ecran, în mod defilare, a programului cuprins între prima linie şi linia curentă.

Oricum, mai întâi se efectuează un calcul aproximativ pentru a vedea cât timp ia listarea şi dacă acesta este prea lung, linia din vârf se mută mai jos pentru a fi mai aproape de linia curentă. Acum, având stabilită linia din vârf, listarea poate începe. Dacă linia curentă a fost listată, listarea se opreşte când s-a ajuns la sfârşitul programului sau la partea de jos a ecranului.

Capitolul 3. LIMBAJUL BASIC

3.1 VARIABILE ŞI EXPRESII ARITMETICE

Cuprins: Nume de variabile, expresii, notaţii
Operaţii: +, -, *, /

Versiunea BASIC a calculatoarelor HC admite pentru variabilele numerice nume formate din oricâte caractere (litere sau cifre), care încep cu o literă. Printre caractere poate fi şi blancul, care este însă ignorat. Prezenţa lui face variabila mai uşor de citit. Sistemul face filtrarea literelor mari, astfel încât atât litera mare cât şi litera mică corespunzătoare sunt interpretate la fel. Nu este indicată folosirea numelor foarte lungi deoarece sunt greu de manipulat.

Variabilele speciale sunt:

1. Variabilele folosite în instrucţiunile **FOR**, care trebuie să fie reprezentate printr-o singură literă.
2. Variabilele de tip şir de caractere, al căror nume este format dintr-o literă urmată de "$".

Expresiile numerice pot fi reprezentate şi printr-un număr zecimal urmat de un exponent.

Exemplul 1. Să se tipărească numerele:

```
PRINT 2.3e0
PRINT 2.34e1
```

şi aşa mai departe până la

```
PRINT 2.34e15
```

Se observă că după un timp calculatorul începe să folosească scrierea cu exponent deoarece nu se pot utiliza mai mult de 14 caractere consecutive pentru scrierea unui număr.

Se poate tipări în mod similar:

```
PRINT 2.34e-1
PRINT 2.34e-2
```

şi aşa mai departe. Comanda **PRINT** afişează numai 8 cifre semnificative.

Exemplul 2.

```
PRINT 4294967295,4294967295-429e7
```


Acest exemplu demonstrează că toate cifrele numărului 4294967295 sunt memorate, deşi nu toate pot fi tipărite pe ecran.

HC-ul utilizează scrierea numerelor în virgulă mobilă.

Numerele sunt reprezentate cu precizie de aproximativ nouă cifre şi jumătate.

Cel mai mare întreg ce poate fi reprezentat cu precizie în memorie este 2e32-1 = 4294967295.

Exemplul 3.

```
PRINT 1e10 + 1-1e10,1e10-1e10 + 1
```

Rezultatele afişate vor fi:

```
0   1
```

deoarece 1e10 + 1 şi 1e10 au aceeaşi reprezentare internă.

Operaţiile aritmetice executate de calculator sunt înmulţirea, împărţirea, adunarea şi scăderea. Operaţiile de înmulţire "*" şi împărţire "/" au prioritate egală. De aceea, o expresie ce conţine numai înmulţiri şi împărţiri se execută de la stânga la dreapta. Adunarea şi scăderea au de asemenea, prioritate egală dar mai mică decât a înmulţirii şi a împărţirii.

Pentru a modifica ordinea de execuţie a operaţiilor se folosesc parantezele.

3.2 ŞIRURI DE CARACTERE

Cuprins: Operaţii cu şiruri de caractere

Şirurile de caractere sunt reprezentate prin secvenţe de caractere ASCII, încadrate între ghilimele ("). Dacă se doreşte tipărirea în text a caracterului ghilimele, el trebuie să fie dublat. Un şir de caractere poate fi atribuit ca valoare unei variabile şir sau poate fi tipărit cu o comandă **PRINT**.

Fiind dat un şir, un subşir al lui constă în câteva caractere consecutive conţinute în el, luate în secvenţă. De exemplu "string" este un subşir al lui "bigger string", însă "b string" nu este. Manipularea subşirurilor în BASIC se face cu:

s(n1 TO n2)

unde:

1. s este un şir de caractere sau o variabilă şir.
2. n1,n2 sunt numere întregi nenegative ce reprezintă ordinul caracterului de început, respectiv de sfârşit, din subşir. Dacă n1« n2, rezultatul este şirul vid ("").

Dacă nu se precizează începutul şi/sau sfârşitul subşirului se iau implicit 1, respectiv lungimea şirului.

Exemplul 1.

"abcdef"(2 TO 5) = "bcde"
"abcdef"(TO 5) = "abcdef"(1 TO 5) = "abcde"
"abcdef"(2 TO) = "abcdef"(2 TO 6) = "bcdef"
"abcdef"(TO) = "abcdef"(1 TO 6) = "abcdef"
"abcdef"(3) = "abcdef"(3 TO 3) = "c"
"abcdef"(5TO7) dă mesaj de eroare deoarece şirul are numai şase caractere
"abcdef"(8 TO 7) = ""
"abcdef"(1 TO 0) = ""

Exemplul 2.

```
10 LET a$ = "Salut Ana !"
20 FOR n = 1 TO 11
30 PRINT a$(n TO 11), a$((12-n) TO 11)
40 NEXT n
50 STOP
```

Exemplul 3.

```
10 LET c$ = "Acesta este un calculator HC"
20 LET c$(13 TO 25) = "hc"
30 PRINT c$
```

După execuţia programului pe ecran va apare mesajul:

Acesta este hc   HC

Dacă într-o atribuire şirul din dreapta conţine mai puţine caractere decât sunt specificate în subşirul din stânga, atunci diferenţa de lungime va fi completată cu blancuri. O astfel de asignare se numeşte "procusteană".

3.3 TABLOURI

Cuprins: Tablouri de numere şi şiruri
DIM

În limbajul BASIC al calculatoarelor HC se pot defini variabile de tip tablou cu oricâte dimensiuni. Elementele tabloului pot fi numere reale, caz în care numele variabilei este reprezentat printr-o singură literă, sau de tip şir de caractere, numele variabilei fiind format dintr-o literă urmată de $. Înainte de a utiliza un tablou, trebuie rezervat spaţiu în calculator pentru el; aceasta se realizează utilizând instrucţiunea **DIM**, a cărei formă este:

DIM m(n1,n2,..,nk)

unde:

1. m - este numele unei variabile de tip tablou.
2. n1,n2,..,nk - sunt numerele maxime de componente corespunzătoare fiecărei dimensiuni a tabloului.

Printr-o comandă **DIM** poate fi definită numai o singură variabilă de tip tablou. Această instrucţiune are următorul efect:

1. rezervă spaţiul necesar tabloului definit.
2. iniţializează elementele tabloului cu 0.
3. şterge orice tablou care are acelaşi nume cu variabila definită prin instrucţiunea curentă.

Se menţionează că pot coexista un tablou şi o variabilă simplă cu acelaşi nume, fără să apară confuzii.

Şirurile dintr-un tablou diferă de şirurile simple prin aceea că au lungime fixă şi asignarea lor este procusteană. Un alt mod de interpretare al unui tablou de şiruri de caractere este ca tablou de caractere simple cu numărul dimensiunilor majorat cu 1 faţă de cazul precedent. Un tablou de şiruri şi o variabilă şir simplă nu pot avea acelaşi nume (spre deosebire de cazul variabilelor numerice).

Pentru a defini un tablou a$ de 5 şiruri, trebuie stabilită mai întâi lungimea şirului - spre exemplu 10 caractere.

Linia:

DIM a$(5,10)

defineşte 5*10 = 50 caractere, dar fiecare rând poate fi interpretat ca un şir.
De exemplu a$(1) este format din:

a$(1,1) a$(1,2) ... a$(1,10)

Dacă sunt utilizate două dimensiuni, se obţine un singur caracter, dar dacă este omisă a doua dimensiune, atunci se obţine un şir cu lungime fixă. Astfel a$(2,7) e al şaptelea caracter în şirul a$(2); o altă notaţie a aceluiaşi element este a$(2)(7).

Ultimul indice poate avea şi forma unui selector de subşir. De exemplu, dacă a$(2) = "12345667890", atunci

a$(2,4 TO 8) = a$(2)(4 TO 8) = "45678"

Se pot defini variabile de tip tablou de şiruri de caractere cu o singură dimensiune; în acest caz variabila se comportă ca o variabilă simplă cu excepţia faptului că are totdeauna lungime fixă iar asignarea ei este procusteană.

Exemplu

DIM a$(10)

3.4 INIȚIALIZAREA VARIABILELOR

Cuprins: READ, DATA, RESTORE

Introducerea constantelor într-un program se face prin grupul de instrucțiuni **READ**, **DATA** și **RESTORE**. Forma generală a unei instrucțiuni **READ** este:

```
READ n1,n2,...
```

unde n1,n2,... este lista variabilelor care trebuiesc inițializate, ele fiind separate prin virgulă. Instrucțiunea **READ** lucrează la fel cu instrucțiunea **INPUT**, exceptând faptul că valorile variabilelor sunt luate dintr-o instrucțiune **DATA**, nu de la terminal.

Fiecare instrucțiune **DATA** este o listă de expresii numerice sau de tip șir de caractere, separate prin virgulă. Instrucțiunile **DATA** pot fi puse oriunde în program, ele comportându-se ca o listă unică realizată prin concatenarea tuturor instrucțiunilor **DATA** din program (lista **DATA**).

Când calculatorul citește prima variabilă cu **READ**, ci îi este asociată prima valoare din lista DATA, și așa mai departe. Dacă se încearcă citirea mai multor variabile decât numărul valorilor din lista **DATA**, atunci apare eroare.

Este posibil să se facă salturi în lista DATA, utilizând instrucțiunea **RESTORE**. Forma instrucțiunii este:

```
RESTORE n
```

Ea face ca instrucțiunea **READ** următoare să citească datele de la o instrucțiune **DATA** aflată la linia "n" sau după aceasta. Dacă "n" lipsește, se ia valoarea implicită 1.

Exemplul 1.

```
10 READ a,b,c
20 PRINT a,b,c
30 DATA 10,20,30
40 STOP
```

Rezultatele programului vor fi:

```
10  20    (a = 10, b = 20)
30        (c = 30)
```

Exemplul 2.

```
10 READ d$
20 PRINT "Data este: ",d$
30 DATA "10 martie 1992"
```


Rezultatul acestui program este:

```
Data este: 10 martie 1992
```

Exemplul 3.

```
10 READ a,b
20 PRINT a,b
30 RESTORE 10
40 READ x,y,z
50 PRINT x,y,z
60 DATA 1,2,3
70 STOP
```

Rezultatele furnizate de acest program sunt:

```
1   2    (a = 1, b = 2)
1   2    (x = 1, y = 2)
3        (z = 3)
```

3.5 OPERAȚII LOGICE

Cuprins: = , « , », « =, » =, « »
AND, OR, NOT

Operațiile aritmetice executate de calculator sunt înmulțirea, împărțirea, adunarea și scăderea. Operațiile de adunare și scădere au prioritate egală dar mai mică decât a înmulțirii și a împărțirii.

Pentru șirurile de caractere s-a definit operația de concatenare, notată cu " + ".

Exemplul 1.

```
10 LET n$ = "Ionescu "
20 LET p$ = "Ana"
30 LET s$ = n$ + p$
40 PRINT s$
50 STOP
```

Programul prezentat va determina tipărirea pe ecran a textului:

```
Ionescu Ana
```

care reprezintă valoarea variabilei s$.

Relațiile de ordine în mulțimea numerelor sunt relațiile de egalitate și de inegalitate apelabile folosind notațiile " = ", "«", "»", "« = ", "» = ", "« »".

În mulțimea șirurilor de caractere relația de ordine folosită este ordonarea alfabetică, relațiile folosite fiind aceleași ca la numere.

Pentru realizarea unor expresii complexe se pot utiliza şi operaţiile logice "OR", "AND" şi "NOT" care admit operanzi de tip boolean. De exemplu instrucţiunea:

IF a$ = "DA" AND x<>0 THEN PRINT x

tipăreşte valoarea numărului "x" dacă sunt îndeplinite simultan cele 2 condiţii.

Similar se pot realiza expresii cu "OR" dacă se doreşte identificarea situaţiei în care cel puţin una dintre condiţii este îndeplinită. Operaţia "NOT" produce ca rezultat inversul valorii argumentului său.

Operaţiile "OR", "AND", "NOT" pot fi aplicate şi unor argumente numerice. Funcţiile definite astfel sunt:

1. x AND y ia valoarea
   x , dacă y e nenul
   0 , dacă y = 0

2. x OR y ia valoarea
   1 , dacă y e nenul
   x , dacă y = 0

3. NOT x ia valoarea
   0 , dacă x e nenul
   1 , dacă x = 0

În continuare sunt prezentate operaţiile recunoscute de limbajul BASIC în ordinea crescătoare a priorităţilor:

1. "OR"
2. "AND"
3. "NOT"
4. relaţiile conditionale
5. "+", "-"
6. "*", "/"

3.6 FUNCŢII

Cuprins: ↑, PI, EXP, LN, SIN, COS, TAN, ASN, ACS, ATN, DEF, LEN, STR$, VAL, SGN, ABS, INT, SQR, FN

Funcţiile definite de calculator au prioritate mai mare decât operaţiile. Dacă în evaluarea unei expresii este necesară o altă ordine de execuţie a operaţiilor şi funcţiilor decât cea determinată de priorităţile lor, atunci se folosesc paranteze.

Funcţiile matematice definite în BASIC sunt ridicarea la putere, funcţia exponenţială, funcţia logaritmică şi funcţiile trigonometrice.

Funcţia ridicare la putere "↑" are prioritate mai mare decât înmulţirea şi împărţirea. Ea necesită 2 operanzi dintre care primul este obligatoriu pozitiv. Într-o

înşiruire de ridicări la putere, ordinea evaluării este de la stânga la dreapta, ceea ce înseamnă că :

$2\uparrow3\uparrow2 = 8\uparrow2 = 64$

Funcţia **EXP** defineşte funcţia exponenţială:

$EXP\ x = e^x$

unde e = 2,71...

Funcţia **LN** calculează logaritmul natural al argumentului.
Ea poate fi utilizată la calculul unui logaritm în orice bază folosind formula:

$LOG_a\ x = LN\ x / LN\ a$

SIN, COS, TAN, ASN, ACS, ATN sunt mnemonicele funcţiilor sinus, cosinus, tangenta, arcsinus, arccosinus şi respectiv arctangenta.

Sistemul pune la dispoziţia utilizatorului numărul "pi", ce poate fi apelat apăsând tasta PI. Comanda **PRINT PI** tipăreşte valoarea numărului "pi".

Funcţiile descrise în continuare sunt disponibile în modul de lucru extins. Acţionarea simultană a tastelor **CAPS SHIFT** şi **SYMBOL SHIFT** determină trecerea din modul "L" în modul "E".

Funcţia **LEN** dă lungimea unui şir.

Exemplul 1.

```
PRINT LEN "majuscule"
```

va determina tipărirea numărului 9.

Funcţia **STR$** converteşte numere în şiruri. Argumentul este un număr, iar rezultatul este şirul care ar apare pe ecran dacă numărul ar fi afişat cu **PRINT**. Se observă că numele funcţiei se sfârşeşte cu "$" pentru a arăta că rezultatul ei este un şir.

Exemplul 2.

```
LET a$ = STR$ 1e2
```

Instrucţiunea de mai sus are acelaşi efect cu:

```
LET a$ = "100"
```

Comanda

```
PRINT LEN STR$ 100.000
```

produce răspunsul 3, deoarece STR$ 100.000 = "100".

Funcţia VAL converteşte şiruri de caractere în numere.

VAL "3.5" = 3.5

Dacă se aplică funcţiile **STR$** şi **VAL** asupra unui număr, totdeauna se va obţine numărul iniţial, pe când dacă se aplică **VAL** urmat de **STR$** asupra unui şir de caractere nu se obţine totdeauna şirul iniţial. Evaluarea funcţiei **VAL** se face în 2 paşi:

1. argumentul este evaluat ca şir.
2. ghilimelele sunt îndepărtate şi caracterele rămase sunt evaluate ca numere.

Exemplul 3.

VAL "2*3" = 6
VAL ("2" + "*3") = 6

Altă funcţie similară lui **VAL** dar mai puţin utilizată este **VAL$**. Şi această funcţie se evaluează tot în 2 paşi; primul pas este la fel cu al funcţiei VAL, dar după înlăturarea ghilimelelor caracterele sunt evaluate ca alt şir.

VAL$ """fructe""" = "fructe"

Funcţia **SGN** aplicată asupra variabilei x are următoarea definiţie:

1. 1, dacă x0
2. 0, dacă x = 0
3. -1, dacă x

Funcţia **ABS** produce valoarea absolută a numărului pe care-l are ca argument.

ABS -3.2 = ABS 3.2 = 3.2

Funcţia **INT** furnizează partea întreagă a argumentului său.

INT 3.9 = 3
INT -3.9 = -4

Funcţia **SQR** calculează rădăcina pătrată a argumentului său care este un număr pozitiv.

SQR 0.25 = 0.5
SQR -4 generează mesaj de eroare

Sistemul permite definirea de funcţii utilizator. Numele posibile pentru acestea sunt **FN** urmat de o literă (dacă rezultatul e un număr), sau **FN** urmat de o literă şi $ (dacă rezultatul e un şir). Obligatoriu argumentul trebuie să fie inclus în paranteze.

Definirea funcţiilor utilizator se face cu funcţia predefinită **DEF**. Definirea funcţiei de ridicare la pătrat se poate face astfel:

DEF FN s(x) = x*x

Rotunjirea unui număr real la cel mai apropiat întreg poate fi făcută prin aplicarea funcţiei **INT** asupra argumentului mărit cu 0.5:

```
20 DEF FN r(x) = INT(X + 0.5)
```

Exemplul 5.

```
10 LET x = 0: LET y = 0: LET a = 10
20 DEF FN p(x,y) = a + x*y
30 DEF FN q() = a + x*y
40 PRINT FN p(2,3), FN q()
```

Când este evaluată FN p(2,3), "a" are valoarea 10, deoarece e variabilă liberă, x are valoarea 2 deoarece este primul argument şi y are valoarea 3 deoarece este al doilea argument. Rezultatul este 10 + 2*3 = 16.

Când este evaluată funcţia fără argumente FN q, a,x şi y sunt variabile libere şi au valorile: 10, 0 respectiv 0. Răspunsul în acest caz este 10 + 0*0 = 10.

Schimbând linia 20 cu

```
20 DEF FN p(x,y) = FN q()
```

de această dată FN p(2,3) va avea valoarea 10.

O funcţie poate avea până la 26 argumente numerice şi în acelaşi timp până la 26 argumente de tip şir de caractere.

3.7 DECIZII

Cuprins: IF, THEN, STOP

Instrucţiunea care realizează luarea deciziilor este de forma:

n IF condiţie THEN comenzi

unde

1. "n" este numărul liniei.
2. "comenzi" este o secvenţă de instrucţiuni care trebuie să fie executată în cazul în care "condiţia" este adevărată.
3. "condiţie" este o relaţie operaţională care în urma evaluării poate fi adevărată sau falsă. Dacă condiţia este adevărată, atunci se execută secvenţa de

instrucţiuni scrisă după THEN. Altfel, programul execută instrucţiunile de pe linia următoare.

Cele mai simple condiţii compară două numere sau două şiruri de caractere. Ele pot testa dacă două numere sunt egale sau dacă unul este mai mare decât celălalt. Se poate testa şi egalitatea a două şiruri de caractere, sau dacă în ordinea alfabetică unul apare înaintea celuilalt.

Exemplu:

```
10 REM Ghiciti numarul
20 INPUT a : CLS
30 INPUT "Ghiciti numarul", b
40 IF b = a THEN PRINT "Rezultat corect" : STOP
50 IF bA THEN PRINT "Prea mic! Mai incearca o data!"
60 IF ba THEN PRINT "Prea mare! Mai incearca o data!"
70 GO TO 30
```

În acest program linia 40 compară variabilele a şi b. Dacă sunt egale, programul este oprit cu comanda **STOP**. În partea de jos a ecranului apare mesajul

```
9 STOP statement, 40:3
```

care arată că oprirea programului este cauzată de a treia instrucţiune din linia 40. Linia 50 determină dacă b este mai mic decât a, iar linia 60 opusul, adică dacă b este mai mare decât a. Instrucţiunea **CLS** din linia 20 şterge ecranul şi împiedică adversarul de joc să vadă ce număr s-a introdus.

3.8 ITERAŢII

Cuprins: FOR, NEXT, TO, STEP

În BASIC instrucţiunea de ciclare este **FOR - NEXT**. Forma generală a instrucţiunii FOR este:

```
FOR v = vi TO vf STEP p
    corp ciclu
NEXT v
```

unde

1. "v" este o variabilă contor specifică ciclului FOR - NEXT; ea trebuie să aibă numele format dintr-o singură literă.
2. "vi" este valoarea cu care este iniţializat contorul ciclului.
3. "vf" este valoarea maximă la care poate ajunge "v"; deci "v" ≪ "vf" (s-a presupus că "p" ≫0).
4. "p" este mărimea pasului; el reprezintă diferenţa între două valori succesive ale contorului.

5. "corp ciclu" este secvenţa de instrucţiuni ce se repetă. "vi","vf" şi "p" pot fi exprimate prin constante, variabile sau expresii de tip real.

În cazul în care "p" este negativ, regula de rămânere în ciclu este "v" = "vf".

Două cicluri FOR - NEXT pot fi imbricate sau complet separate. Este greşită suprapunerea parţială a două cicluri. De asemenea trebuie evitat saltul din exterior în interiorul unei bucle FOR - NEXT deoarece contorul nu poate fi iniţializat decât printr-o instrucţiune FOR. Pentru a fi siguri că nu se fac salturi în interiorul unui ciclu, se pot scrie toate instrucţiunile ciclului pe o singură linie (dacă spaţiul permite).

Exemplul 1.

```
10 FOR n = 10 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

Exemplul 2.

```
50 FOR m = 0 TO 6
60 FOR n = 0 TO m STEP 1/2
70 PRINT m;":";n;" ";
80 NEXT n
90 PRINT
100 NEXT m
```

Exemplul 3.

```
100 FOR m = 0 TO 10: PRINT m: NEXT m
```

Exemplul 4.

```
FOR n = 0 TO 1 STEP 0: INPUT a: PRINT a: NEXT n
```

Această comandă determină repetarea la infinit a instrucţiunii **INPUT** în modul de lucru imediat (deci nu prin program). Dacă apare o eroare, comanda **INPUT** se pierde şi deci pentru continuarea citirii trebuie rescrisă întreaga linie.

3.9 SUBRUTINE

Cuprins: GOSUB, RETURN

Utilizarea subrutinelor este posibilă prin utilizarea instrucţiunii **GOSUB** (go to subroutine-apel de subrutină) şi **RETURN** (revenire din subrutină). Instrucţiunea **GOSUB** are forma:

```
GO SUB n
```

unde "n" este numărul primei linii din subrutină. Ea este asemănătoare instrucțiunii **GO TO n**, cu excepția faptului că în cazul instrucțiunii **GOSUB** este memorată adresa instrucțiunii, astfel încât după executarea subrutinei, programul continuă cu instrucțiunea următoare saltului la subrutină. Aceasta se realizează memorând numărul liniei și numărul instrucțiunii din linie (care împreună formează adresa de revenire) într-o stivă.

Instrucțiunea **RETURN** ia adresa din vârful stivei **GOSUB** și merge la instrucțiunea care îi urmează.

În BASIC subrutinele sunt recursive.

Exemplu:

```
10 INPUT a: CLS
20 INPUT "ghiciti numarul !",b
30 IF a=b THEN PRINT "corect !!!": STOP
40 IF a THEN GO SUB 90
50 IF ab THEN GO SUB 90
60 GO TO 20
90 PRINT "Mai incearca o data !"
100 RETURN
```

Instrucțiunea **GO TO** este foarte importantă deoarece sistemul semnalează eroarea dacă, în execuție, întâlnește un **RETURN** care nu a fost precedat de un **GO SUB.**

3.10 GENERAREA NUMERELOR ALEATOARE

Cuprins: RND, RANDOMIZE

Generarea numerelor aleatoare se face cu funcția predefinită **RND.** Ea nu este o funcție complet aleatoare ci o funcție periodică cu perioada suficient de mare (65535), astfel încât efectul de periodicitate poate fi neglijat. În cadrul unei perioade, numerele generate sunt complet aleatoare. În anumite privințe, **RND** se comportă ca o funcție fără argumente: efectuează calcule și produce un rezultat. De fiecare dată când e utilizată, rezultatul său este un număr aleator nou, cuprins între 0 și 1 (uneori poate lua valoarea 0, dar niciodată 1). Dacă se dorește ca numerele aleatoare să fie într-un anumit domeniu de valori se poate proceda ca în exemplele următoare:

```
5*RND            genereaza numere intre 0 si 5;
1.3+0.7*RND      produce numere intre 1.3 si 2;
1+INT(RND*6)     furnizeaza numere aleatoare intregi intre 1 si 6.
```

Exemplu


```
10 REM Program de simulare a aruncarii zarurilor
20 CLS
30 FOR n=1 TO 2
40 PRINT 1+INT(RND*6);" ";
50 NEXT n
60 INPUT a$: GO TO 20
```

Linia 60 face să fie generată o pereche de numere aleatoare după fiecare apăsare a tastei **CR**.

Funcția **RANDOMIZE** e utilizată pentru a face ca **RND** să pornească dintr-un punct definit al secvenței de numere; argumentul său este un număr între 1 și 65535 care reprezintă numărul de ordine al viitorului apel al funcției **RND**. Efectul instrucțiunii **RANDOMIZE** se poate vedea în programul următor.

```
10 RANDOMIZE 1
20 FOR n=1 to 5 :PRINT RND :NEXT n
30 PRINT:GO TO 10
```

După fiecare execuție a instrucțiunii **RANDOMIZE 1, RND** va furniza o secvență de 5 numere ce începe cu 0.0022735596, care este primul număr generat de funcția **RND** (are numărul de ordine 1). **RANDOMIZE** poate fi folosit la testarea programelor ce conțin funcția **RND**, deoarece secvența numerelor aleatoare generate este mereu aceeași.

RANDOMIZE, ca și **RANDOMIZE 0**, are efect diferit de **RANDOMIZE** urmat de un număr. Această instrucțiune utilizează timpul trecut de la punerea în funcțiune a calculatorului. Programul:

```
10 RANDOMIZE
20 PRINT RND: GO TO 10
```

determină tipărirea aceluiași număr. Deoarece timpul de lucru al calculatorului a crescut cu aceeași cantitate la fiecare execuție a lui **RANDOMIZE**, următorul **RND** furnizează aproximativ același rezultat.

Pentru a se obține o secvență aleatoare se înlocuiește GO TO 10 cu GO TO 20.

Exemplu

Programul determină frecvența de apariție a "capului" și a "pajurei" la aruncarea unei monezi.

```
10 LET cap=0:LET pajura=0
20 LET moneda=INT(RND*2)
30 IF moneda=0 THEN LET cap=cap+1
40 IF moneda=1 THEN LET pajura=pajura+1
50 PRINT cap; ",";pajura
60 IF pajura0 THEN PRINT cap/pajura;
70 PRINT: GO TO 20
```

Dacă timpul de rulare este suficient de mare, raportul cap/pajura devine aproximativ 1, deoarece numerele aleatoare generate sunt uniform repartizate în intervalul 0,1.

3.11 SETUL DE CARACTERE

Cuprins: CODE, CHR$, POKE, PEEK, USR, BIN

Alfabetul utilizat de HC cuprinde 256 caractere şi fiecare are un cod între 0 şi 255. Caracterele pot fi simboluri simple sau cuvinte cheie ca **PRINT**, **STOP**, , etc.

Pentru conversia între coduri şi caractere, limbajul posedă două funcţii: **CODE** şi **CHR$**. **CODE** se aplică unui şir şi întoarce codul primului caracter al şirului (sau 0 dacă şirul e vid).

CHR$ se aplică unui număr şi produce caracterul ce are acel cod.

Setul de caractere este format din: caracterele ASCII, cuvinte cheie, caractere grafice definite de utilizator.

Un caracter se desenează pe o reţea de 8*8 puncte, fiecărui punct corespunzându-i un bit în memorie. Pentru programarea unui caracter definit de utilizator este necesară descrierea stării fiecărui punct al matricii prin care se reprezintă caracterul respectiv:

1. 0 corespunde unui punct alb (hârtie)
2. 1 corespunde unui punct negru (cerneală)

Pentru definirea caracterului se folosesc 8 instrucţiuni **BIN**. O instrucţiune **BIN** descrie o linie a caracterului, argumentul său fiind format din 8 cifre binare.

Cele 8 numere sunt memorate în 8 octeţi care corespund aceluiaşi caracter.

Instrucţiunea **USR** converteşte un argument de tip şir în adresa din memorie a primului octet al caracterului definit de utilizator corespunzător argumentului. Argumentul trebuie să fie un singur caracter; el poate fi graficul definit de utilizator sau litera corespunzătoare (majusculă sau minusculă).

POKE memorează un număr direct într-o locaţie de memorie, fără să facă apel la mecanismele utilizate în mod obişnuit în BASIC. Opusul lui **POKE** este **PEEK**, care ne permite să vizualizăm conţinutul unei locaţii de memorie, fără a-l modifica.

Pentru a defini caracterul grafic pi (care să apară pe ecran la apăsarea tastei **P** în mod grafic) se utilizează următoarea secvenţă de program:

```
10 FOR n = 0 TO 7
20 INPUT acum: POKE USR "p" + n, acum
30 NEXT n
```

Datele introduse vor fi (în ordinea prezentată):

```
BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
```



```
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000
```

În cele ce urmează se prezintă modul de obţinere a cuvintelor cheie. Caracterele 0,..,31 sunt caractere de control al modului de lucru. De exemplu **CHR$6** realizează tabularea pe orizontală (efect similar unei virgule într-o instrucţiune **PRINT**).

```
PRINT 1; CHR$ 6; 2
```

are acelaşi efect cu:

```
PRINT 1,2
```

şi cu:

```
LET a$ = "1" + CHR$6 + "2"
PRINT a$
```

CHR$8 determină mutarea cursorului înapoi cu o poziţie.

Exemplu:

```
PRINT "1234"; CHR$8; "5"
```

tipăreşte:

```
1235
```

CHR$13 mută cursorul la începutul liniei următoare.

Utilizând codurile pentru caractere putem extinde conceptul de ordine alfabetică pentru a acoperi şiruri ce conţin orice caractere, nu numai litere, folosind în locul alfabetului uzual de 26 litere, alfabetul extins de 256 caractere (la codificarea caracterelor s-a avut în vedere că ordinea crescătoare a codurilor ataşate literelor să coincidă cu ordinea alfabetică).

Este prezentată mai departe o regulă de găsire a ordinii în care se află două şiruri. Mai întâi se compară primele caractere. Dacă sunt diferite, unul dintre ele are codul mai mic decât celălalt şi, deci, se poate decide care este ordinea alfabetică a şirurilor. Dacă aceste coduri sunt egale, se compară următoarele caractere.

Exemplu

```
5 LET b = BIN 01111100:LET c = BIN 00111000:LET d = BIN
  00010000
10 FOR n = 1 TO 6: READ p$: REM 6 piese
20 FOR f = 0 TO 7: REM citeste piesele în octeti
30 READ a: POKE USR p$ + f,a
```

```
40 NEXT f
50 NEXT n
100 REM bishop
110 DATA "b", 0, 0, BIN 001001000, BIN 01000100
120 DATA BIN 01101100, c, b, 0
130 REM king
140 DATA "k", 0, d, c, d
150 DATA c, BIN 010001000, c, 0
160 REM rook
170 DATA "r", 0, BIN 01010100, b, c
180 DATA c,b,b,0
190 REM queen
200 DATA "q",0,BIN 01010100, BIN 00101000, d
210 DATA BIN 01101100, b, b, 0
220 REM pawn
230 DATA "p", b, 0, d, c
240 DATA c, d, b, 0
250 REM knight
260 DATA "n", 0, d, c, BIN 01111000
270 DATA BIN 00011000, c, b, 0
```

3.12 GRAFICE

Cuprins: PLOT, DRAW, CIRCLE, POINT

În acest capitol se prezintă trasarea desenelor cu HC-ul. Partea utilizabilă a ecranului are 22 de linii și 32 de coloane (22*32 = 704 poziții de caractere). Fiecare poziție de caracter e un pătrat format din 8*8 puncte. Punctele se numesc pixeli (picture elements). Un pixel se specifică prin coordonatele sale. Coordonata "x" arată distanța față de extrema stânga, iar coordonata "y" reprezintă distanța față de baza ecranului. Coordonatele se scriu de obicei ca o pereche de numere, în paranteze. Astfel (0,0), (255,0), (0,175), (255,175) sunt extremele stânga jos, dreapta jos, stânga sus, dreapta sus.

Instrucțiunea

```
PLOT    x,y
```

desenează punctul de coordonate x,y.

Programul:

```
10 PLOT INT (RND *256), INT(RND *175):INPUTa$:GO TO 10
```

scrie aleator un punct pe ecran de fiecare dată când se acționează **CR**. Programul următor trasează graficul funcției SIN pentru valori între 0 și 2*pi.

```
10 FOR n = 0 TO 255
20 PLOT n,88 + 80*SIN(n/128*pi)
```



```
30 NEXT n
```

Calculatorul desenează linii drepte, cercuri și porțiuni de cerc utilizând instrucțiunile **DRAW** și **CIRCLE**. Cu

```
DRAW x,y
```

se poate trasa o linie dreaptă. Linia începe din punctul în care se află cursorul ultimei instrucțiuni **PLOT, DRAW**, sau **CIRCLE**. Comenzile **RUN, CLEAR, CLS** și **NEW** îl resetează, aducându-l pe poziția (0,0).

DRAW determină lungimea și direcția liniei. De remarcat că argumentele unei instrucțiuni **DRAW** pot fi și negative.

```
PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35
```

Calculatorul HC are facilități pentru a desena în culori. Următorul program demonstrează acest lucru:

```
10 BORDER 0: PAPER 0: INK 7: CLS: REM tot ecranul este negru
20 LET x1 = 0: LET y1 = 0: REM inceputul liniei
30 LET c = 1: REM prima culoare cu care se deseneaza este albastru
40 LET x2 = INT(RND*255): LET y2 = INT(RND*176):
  REM capatul liniei este aleator
50 DRAW INK c; x2-x1,y2-y1
 60 LET x1 = x2: let y1 = y2: REM urmatoarea linie incepe de unde s-a
terminat precedenta
70 LET c = c + 1: IF c = 8 THEN LET c = 1: REM alta culoare
80 GO TO 40
```

Comenzile **PAPER, INK, FLASH, BRIGHT, INVERSE, OVER** pot apare în instrucțiuni **PLOT** sau **DRAW** în același fel în care apar în **PRINT** și **INPUT**.

Comanda **DRAW** permite și trasarea de porțiuni de cercuri. Forma generală este:

```
DRAW x,y,a
```

unde x,y semnifică punctul final al liniei iar a este numărul de radiani corespunzător circumferinței. Când a este pozitiv porțiunea de cerc se trasează în sens antiorar în timp ce, pentru a negativ se desenează în sens orar. Pentru a = pi se trasează un semicerc, indiferent de valorile luate de x și y (raza este funcție de punctul inițial și de cel final):

```
10 PLOT 100,100: DRAW 50,50,pi
```

Trasarea cercurilor se face cu o comandă **CIRCLE** a cărei formă este:

```
CIRCLE x,y,r
```

unde r este raza cercului iar (x,y) sunt coordonatele centrului cercului. Ca și instrucțiunile **PLOT** și **DRAW**, și **CIRCLE** admite comenzi de modificare a culorii.

Funcția **POINT** arată dacă un pixel are asociată culoarea **INK** sau culoarea **PAPER**. Ea are două argumente numerice care reprezintă coordonatele pixel-ului care trebuie să fie închis între paranteze. Rezultatul este:

1. 0 - dacă punctul are culoarea fundalului (paper).
2. 1 - dacă are culoarea INK.

```
CLS: PRINT POINT (0,0): PLOT 0,0:  PRINT POINT(0,0)
```

Se scrie

```
PAPER 7: INK 0
```

Într-o instrucțiune **PLOT x,y**, **REVERSE** și **OVER** afectează doar pixel-ul desemnat, nu și restul pozițiilor din caracter. Deoarece aceste comenzi sunt în mod normal dezactivate (0), pentru a le activa (1), trebuiesc incluse într-o comandă **PLOT**.

Se poate face ca punctul (x,y) să ia culoarea "ink" prin

```
PLOT x,y;
```

```
PLOT INVERSE 1;
```

face ca pixel-ul (x,y) să ia culoarea fundalului;

```
PLOT OVER 1; x,y
```

inversează culoarea pixel-ului specificat.

```
PLOT INVERSE 1; OVER 1; x,y
```

lasă pixel-ul nemodificat dar schimbă poziția de tipărire.

Alt exemplu de utilizare al instrucțiunii **OVER** este următorul:

-se umple ecranul scriind negru pe alb și apoi se tastează:

```
PLOT 0,0:  DRAW OVER 1,255,175
```

-se trasează astfel o linie (cu întreruperi acolo unde traversează caracterele tipărite pe ecran).

-reexecutând comanda, linia trasată anterior o să dispară.

Avantajul instrucțiunii **OVER** este că permite să se deseneze și apoi să se șteargă desenele fără a afecta ce se află anterior pe ecran.

Utilizând programul

```
PLOT 0,0: DRAW 255,175
```



```
PLOT 0,0:  DRAW INVERSE 1;  255,175
```

se constată că această comandă șterge și părțile din caracterele tipărite anterior.

Dacă se scrie o linie cu:

```
PLOT 0,0:  DRAW OVER 1;  250,175
```

se constată că ea nu va putea fi ștearsă cu:

```
DRAW OVER 1;-250,-175
```

deoarece parcurgerea dreptei într-un sens și în celălalt nu se face exact prin aceleași puncte. O linie se șterge pe aceeași direcție și în același sens în care a fost trasată.

Pentru a extinde gama de culori se amestecă două culori de bază pe un singur pătrat, folosind un caracter grafic definit de utilizator. Programul următor definește un caracter grafic echivalent unei table de șah.

```
1000 FOR n = 0 TO 6 STEP 2
1010 POKE  USR  "a" + n,  BIN  01010101: POKE USR
     "a" + n + 1, BIN 10101010
1020 NEXT n
```

3.13 INSTRUCȚIUNI DE INTRARE-IEȘIRE

Cuprins: PRINT,INPUT
Utilizarea separatorilor :, ;, „ TAB, AT, LINE, CLS

Expresiile folosite pentru a tipări valori cu instrucțiunea **PRINT** sunt numite elementele instrucțiunii și sunt separate între ele cu virgulă sau punct și virgulă (separatori). Un element al instrucțiunii **PRINT** poate lipsi și în acest caz pot apare 2 virgule, una dupa alta.

Există 2 elemente ale instrucțiunii **PRINT** care servesc la poziționarea cursorului în vederea tipăririi. Acestea sunt **AT** și **TAB**.

```
AT linie,coloană
```

deplasează cursorul (locul unde va fi tipărit următorul element) la linia și la coloana specificate. Liniile sunt numerotate de la 0 la 21 (de sus în jos) și coloanele de la 0 la 31 (de la stânga la dreapta).

Exemplu

```
PRINT AT 11,16;"*"
```

imprimă un asterisc în centrul ecranului. Instrucțiunea

TAB coloana

deplasează cursorul în coloana specificată. **TAB** determină deplasarea pe aceeaşi linie pe care se găseşte cursorul, exceptînd cazul când poziţia de tipărire specificată se află înaintea poziţiei de tipărire actuală; în această situaţie se face o deplasare la linia următoare.

Obs. :calculatorul consideră coloanele din instrucţiunea TAB "modulo 32" (adică TAB 33 este echivalent cu TAB 1).

Exemplul de mai jos arată cum se poate tipări începutul paginii 1 a unei cărţi:

```
PRINT TAB 30;1;TAB 12; "Index"; AT 3,1;
"Capitol"; TAB 24; "Pagina"
```

Un exemplu din care rezultă reducerea modulo 32 a numărului din instrucţiunea **TAB** este următorul:

```
10 FOR n=0 TO 20
20 PRINT TAB 8*n;n;
30 NEXT n
```

De reţinut următoarele observaţii:

1. Elementele de tipărire care urmează instrucţiunilor **TAB** sau **AT** sunt de obicei terminate cu ";". Dacă s-ar folosi "," sau nimic, cursorul, după ce este poziţionat, se deplasează.
2. Liniile 22 şi 23 ale ecranului nu pot fi folosite pentru tipărire. Ele sunt rezervate pentru comenzi, pentru citirea datelor, mesaje, etc.
3. Tipărind cu **AT** într-o poziţie deja scrisă, ultima tipărire o anulează pe precedenta.

CLS şterge tot ecranul, funcţie care mai este realizată şi de comenzile **CLEAR** şi **RUN** (care mai execută şi alte funcţii). Când calculatorul, în timp ce tipăreşte, ajunge la ultima linie a ecranului, execută "scrolling" anulând prima linie.

Exemplu:

```
CLS: FOR n=1 TO 22: PRINT n: NEXT n
```

şi apoi,

```
PRINT 99
```

de mai multe ori.

În timpul tipăririi, după ce calculatorul a umplut complet ecranul, se opreşte scriind în partea de jos:

scroll ?

Se răspunde cu "y" sau "n".

Instrucţiunea INPUT

O linie de **INPUT** este compusă dintr-o serie de elemente şi de separatori care au aceeaşi funcţie ca într-o linie de **PRINT**. **INPUT** consideră orice element care începe cu o literă ca pe o variabilă asignabilă (căreia urmează să i se introducă valoarea de la tastatură). Instrucţiunea **INPUT** poate tipări şi mesaje; pentru a tipări un şir de caractere este suficientă introducerea acestuia între ghilimele. Dacă conţine şi valori de variabile, mesajul se închide între paranteze.

Dacă se doreşte citirea unei variabile de tip şir de caractere, a$, pe ecran apare caracterul ghilimele. Dacă această variabilă trebuie să ia valoarea unei alte variabile de tip şir definită în program, b$, aceasta se face prin ştergerea ghilimelelor şi introducerea numelui variabilei (b$).

Toate elementele instrucţiunii **PRINT** care nu sunt supuse acestor reguli pot fi elemente ale instrucţiunii **INPUT**.

Exemplu

```
LET virsta mea=INT( RND*100): INPUT ("Eu am";
virsta mea; "ani." ); "citi ani ai ?"; virsta ta
```

Variabila "virsta mea" este conţinută între paranteze, deci valoarea sa se tipăreşte, în timp ce variabila "virsta ta" nu este între paranteze, şi deci valoarea sa se citeşte de la tastatură.

O altă modalitate de citire a variabilelor şir constă în scrierea cuvântului cheie **LINE** după **INPUT** şi înaintea variabilei şir de citit:

```
INPUT LINE a$
```

În acest caz calculatorul nu va tipări ghilimelele, care în mod normal sunt tipărite când se aşteaptă introducerea unei variabile şir, chiar dacă se comportă ca şi cum ar fi fost. Astfel, scriind carte ca variabilă de intrare, a$ va lua valoarea "carte". Deoarece ghilimelele nu sunt tipărite, nu este posibilă introducerea altui şir. De notat că LINE nu poate fi folosit pentru variabile numerice.

Caracterele de control **CHR$22** şi **CHR$23** funcţionează aproape similar lui **AT** şi **TAB**. Caracterul de control pentru **AT** este **CHR$22**. Primul caracter care îl urmează specifică numărul de linie, iar al doilea numărul coloanei, astfel că:

```
PRINT CHR$22 + CHR$1 + CHR$c;
```

este analog lui

```
PRINT AT 1,c;
```

CHR$1 şi **CHR$c** (c = 13) în mod normal au altă semnificaţie, pe care însă şi-o pierd când urmează după CHR$22.

Caracterul de control echivalent lui **TAB** este **CHR$23** şi cele două caractere care-l urmează sunt folosite pentru a indica un număr cuprins între 0 şi 65535 care specifică numărul de **TAB** ca şi argumentul unei instrucţiuni **TAB**.

```
PRINT CHR$23 + CHR$a + CHR$b
```

este echivalent lui

```
PRINT TAB a + 256*b
```

Dacă nu se doreşte afişarea mesajului "scroll ?" la sfârşitul fiecărui ecran, se poate folosi:

```
POKE 23692,255
```

din când în când. După această linie calculatorul inhibă mesajul "scroll ?" pentru următoarele 255 linii.

3.14 CULORI

Cuprins: PAPER, INK, FLASH, INVERSE, OVER, BORDER, ATTR

Calculatorul HC are facilităţi color. El foloseşte 8 culori (numerotate de la 0 la 7). Lista culorilor în ordinea în care sunt pe tastele numerice este următoarea:

0 - negru
1 - albastru
2 - roşu
3 - purpuriu (magenta)
4 - verde
5 - albastru deschis
6 - galben
7 - alb

Într-un televizor alb-negru aceste numere corespund unor tonuri de gri ordonate de la închis spre deschis.

Orice caracter are asociate 2 culori: culoarea caracterului propriu-zis şi culoarea fondului (vezi subcapitolul Setul de caractere). La pornirea calculatorului, sistemul lucrează în alb- negru, cu caractere negre pe fond alb. Tipărirea poate fi făcută normal, dar există şi posibilitatea să apară pe ecran pâlpâind (flash). Pâlpâirea se obţine inversând continuu culoarea caracterului cu culoarea fondului. Deoarece atributele de culoare şi pâlpâire sunt asociate caracterelor (deci matricilor de 64 puncte), nu este posibil ca într-un caracter să fie mai mult de două culori. Valorile acestor atribute pot fi modificate cu instrucţiunile **INK**, **PAPER** şi **FLASH**. Forma acestor instrucţiuni este:


```
PAPER n
INK n
FLASH m
```

unde

1. n este un număr cuprins între 0 şi 7.
2. m este un număr binar (0 pentru inactiv şi 1 pentru activ).

Pentru ilustrarea modului de folosire al instrucţiunilor prezentate se propune programul:

```
20 FOR n=1 TO 10
30 FOR c=0 TO 7
40 PAPER c: PRINT " ";:REM spatii colorate
50 NEXT c: NEXT n
60 PAPER 7
70 FOR c=0 TO 3
80 INK c: PRINT c;" ";
90 NEXT c: PAPER 0
100 FOR c=4 TO 7
110 INK c: PRINT c;" ";
120 NEXT c
130 PAPER 7: INK 0
```

În afară de aceste valori de argumente a căror semnificaţie a fost deja prezentată, mai pot fi folosite valorile 8 şi 9. 8 poate fi folosit ca argument pentru toate cele 4 comenzi şi semnifică transparenţa, fapt ce nu alterează atributele poziţiei la tipărirea unui caracter. De exemplu:

```
PAPER 8
```

face ca la tipărirea unui caracter, culoarea fondului să fie aceeaşi cu a caracterului tipărit anterior. 9 poate fi folosit numai cu comenzile **PAPER** şi **INK** şi indică contrastul. Culoarea "cernelii" sau a "hârtiei" (fundalului), în funcţie de comanda utilizată, este făcută să contrasteze cu cealaltă, punând alb pe o culoare închisă (negru, albastru, roşu, magenta) şi negru pe o culoare deschisă (verde, bleu, galben, alb).

```
INK 9: FOR c=0 TO 7: PAPER c: PRINT c: NEXT c
```

Rulând programul

```
INK 9: PAPER 8: PRINT AT 0,0; FOR n=1 TO 1000:
PRINT n: NEXT n
```

după primul program din acest paragraf, culoarea cernelii este făcută mereu să contrasteze cu vechea culoare pe care o avea fundalul în fiecare poziţie. Comanda

INVERSE 1

inversează fundalul cu cerneala pentru caracterul specificat.

Comanda

OVER 1

realizează supratipărirea. În mod obişnuit, când ceva este scris într-o poziţie de caracter, şterge complet ce era scris înainte; de data aceasta noul caracter va fi doar adăugat. Acest lucru este util în scrierea caracterelor compuse, cum ar fi literele cu accente. Trebuie utilizat în acest scop caracterul de control **CHR$8** pentru întoarcerea cu o poziţie.

Există o altă posibilitate de a utiliza **INK, PAPER, FLASH**. Pot apare în **PRINT** urmate de ";" şi fac exact acelaşi lucru pe care l-ar face când sunt utilizate independent, exceptând faptul că efectul lor este numai temporar.

Astfel dacă se rulează:

```
PRINT PAPER 6; "x";: PRINT "y"
```

numai x va fi pe fond galben.

INK şi celelalte comenzi nu afectează culorile părţii de jos a ecranului. Aceasta foloseşte culoarea marginii drept culoare a fundalului şi codul 9 pentru a contrasta culoarea cernelii. Nu are posibilitatea de pâlpâire şi este cu luminozitate normală.

Marginea poate lua oricare din cele 8 culori (0-7) cu comanda

BORDER culoare

Se pot schimba culorile mesajului scris pe ecran cu comanda **INPUT**, inserând în această comandă **INK, PAPER**, etc, ca şi în cazul comenzii **PRINT**. Efectul lor este activ numai asupra comenzii următoare:

```
INPUT FLASH 1; INK 1; "text"; n
```

Comenzile pot fi schimbate utilizând caracterele de control ca şi în cazul **AT** şi **TAB** (vezi capitolul instrucţiuni de intrare- ieşire).

CHR$16 -- INK
CHR$17 -- PAPER
CHR$18 -- FLASH
CHR$20 -- INVERSE
CHR$21 -- OVER

Aceste caractere de control sunt urmate de un caracter care indică culoarea prin intermediul codului său. De exemplu:

```
PRINT CHR$16 + CHR$9 ; ...
```


are acelaşi efect cu:

```
PRINT INK 9; ...
```

Funcţia **ATTR** are forma:

ATTR (linie,coloana)

Rezultatul este un număr care arată atributele pentru caracterul aflat la linia şi coloana precizată. Numărul este suma a patru numere, conform schemei:

1. 128 - dacă poziţia pâlpâie , 0 dacă este stabilă
2. 64 - dacă poziţia este strălucitoare , 0 dacă este normală
3. 8*n - n = codul fundalului
4. m - m = codul cernelii

Exemplu: Pentru o poziţie pâlpâitoare, normală, cu fundal galben şi cerneala albastră se obţine:

$128+0+8*6+1=177$

3.15 MIŞCAREA

Cuprins: PAUSE, INKEY$, PEEK

Pentru a realiza o pauză în program în timpul căreia nu se desfăşoară nici o operaţie se foloseşte comanda:

PAUSE n

care opreşte execuţia programului menţinînd activ display-ul pe durata a n perioade de baleiaj ale ecranului (20 ms pentru fiecare ecran); n poate lua valoarea maximă 65535, căreia îi corespunde o pauză de aproximativ 22 minute. Dacă n = 0 se opreşte definitiv.

O pauză obţinută în acest mod poate fi scurtată apăsând orice tastă (cu excepţia lui **SPACE** şi **CAPS SHIFT** care produce **BREAK**).

Programul următor desenează cadranul unui ceas pe care se mişcă secundarul:

```
 10 REM Mai intii e desenat cadranul.
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS( n/PI), 16+10*SIN( n/PI)
 40 NEXT n
 50 REM Se porneste ceasul.
 60 FOR t=0 TO 200000; :REM t e timpul în secunde
 70 LET a=t/30*PI: REM a este unghiul secundarului în radiani
 80 LET sx=80*SIN( a): LET sy=80*COS( a)
200 PLOT 128,88: DRAW OVER 1; sx, sy: REM Se desenează secundarul
```

```
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1; sx, sy: REM Se sterge secundarul
230 NEXT t
```

Cu linia 210 se marchează trecerea unei secunde; s-a folosit n = 42 şi nu n = 50 deoarece calculatorul foloseşte un timp pentru scrierea liniilor ciclului **FOR - NEXT**; linia 210 opreşte calculatorul doar pentru timpul care mai ramâne.

O temporizare mai precisă se poate realiza citind conţinutul anumitor locaţii de memorie cu PEEK. Expresia următoare:

(65536 *PEEK 23674 + 256*PEEK 23673 + PEEK 23672)/50

dă numărul de secunde scurse de la aprinderea calculatorului până la 3 zile şi 21 ore, după care se resetează. Programul unui ceas mai precis este dat în continuare:

```
10 REM Se deseneaza cadranul
20 FOR n = 1 TO 12
30 PRINT AT 10-10*cos(n/6*pi),16 + 10*SIN(n/6*PI);n
40 NEXT n
50 DEF FNt() = INT(65536* PEEK 23674 + 256* PEEK
   23673 + PEEK 23672)/50: REM Numarul de secunde de
   la inceput
100 REM se porneste ceasul
110 LET t1 = FNt()
120 LET a = t1/30*PI: REM a este unghiul în radiani
130 LET sx = 72* SIN a: LET sy = 72*COS a
140 PLOT 131,91: DRAW OVER 1; sx; sy: REM
    Se deseneaza secundarul
200 LET t = FNt()
210 IF t = t1 THEN GO TO 200
220 PLOT 131,91: DRAW OVER 1; sx; sy: REM Se
    sterge vechiul secundar
230 LET t1 = t: GO TO 120
```

Acest ceas se opreşte temporar de câte ori se execută **BEEP** ori se utilizează imprimanta, casetofonul. Numerele PEEK 23674, PEEK 23673 şi PEEK 23672 sunt folosite pentru a număra în incremente de 20 ms. Fiecare variază de la 0 la 255, după care se reîncepe. Cel mai rapid se incrementeaza locatia 23672 (cu 1 la fiecare 20 ms); când se trece de la 255 la 0, locaţia 23673 se incrementează cu 1; analog pentru 23674. Presupunând că cele 3 numere sunt 0 (pentru PEEK 23674), 255 (pentru PEEK 23673) şi 255 (pentru PEEK 23672), au trecut deci circa 21 minute de la pornirea calculatorului. Expresia devine:

(65536*0 + 256*255 + 255)/50 = 1310.7

Pentru a poziţiona ceasul pe ora 10 se procedează astfel:

10*60 *60 *50 = 1800000 = 65536*27 + 256*119 + 64

şi se memorează numerele 27, 119 şi 64 cu:

```
POKE 23674,27: POKE 23673,119: POKE 23672,64
```

Funcţia **INKEY$**, fără argument, dă caracterul apăsat pe tastă în momentul apelării sale. Cu programul următor calculatorul devine o maşină de scris:

```
10 IF INKEY$ "" THEN GO TO 10
20 IF INKEY$ = "" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

Linia 10 aşteaptă să se elibereze ultima tastă apasată; linia 20 aşteaptă apăsarea uneia noi. Spre deosebire de **INPUT**, **INKEY$** nu aşteaptă apăsarea lui **CR** sau a unei taste.

3.16 MEMORIA

Cuprins: CLEAR

Fiecărui octet îi este asociată o adresă care este un număr între 0 şi FFFFH. Memoria este împărţită în trei zone distincte:

1. 0 - 4000H zona ROM

în această zonă se găseşte memoria ROM în care este înregistrat interpretorul BASIC.

2. 4000H - 7FFFH zona RAM video

în această zonă se găseşte memoria video cât şi o parte din memoria RAM de program.

3. 8000H - FFFFH zona RAM suplimentar

această zonă nu este neapărat necesară. Ea este folosită pentru mărirea capacităţii de memorie. Ea diferă de zona video printr-un timp de acces mai mic.

Conţinutul memoriei poate fi vizualizat cu funcţia PEEK care are ca argument

ROM	RAM VIDEO	RAM SUPLIMENTAR

Fig. 3.1.

o adresă. Exemplul următor vizualizează primii 21 octeţi din memoria ROM şi adresele lor:

```
10 PRINT "Adresa"; TAB 10; "Octet"
20 FOR a = 0 TO 20
```

```
30 PRINT a TAB 10; PEEK a
40 NEXT a
```

Schimbarea conţinutului memoriei RAM se poate face cu instrucţiunea **POKE**, care are forma:

POKE adresa, conţinut nou

unde "adresa" şi "conţinut nou" sunt expresii numerice.

```
POKE 31000, 57
```

determină încărcarea valorii 57 la adresa 31000. Cu

```
PRINT PEEK 31000
```

se va tipări 57. "Conţinut nou" trebuie să aibă valoarea între -255 şi 255. Dacă e număr negativ, se adună 256.

De importanţă pentru utilizator este organizarea memoriei RAM. Memoria este împărţită în zone specifice stocării unui anumit gen de informaţie. Zonele sunt suficient de mari pentru ca informaţia conţinută actualmente să poată fi reorganizată dacă se inserează ceva într-un anumit punct (de exemplu prin adăugarea unei linii de program sau a unei variabile). La inserare, spaţiul necesar este creat prin mutarea în sus a tot ce se află deasupra. Dacă se şterge informaţie, atunci totul este mutat în jos.

Variabilele sistem (**PROG, CHANS, VARS, ELINE**, etc.) conţin diferite informaţii necesare pentru gestiunea internă a memoriei. Ele indică limitele pentru

Fişier display	Atribute	Buffer imprimanta	Variabile sistem	Harta disc
16384	22528	23296	23552	23734 CHANS

Informaţii de canal	80H	Program BASIC	Variabile	80H
CHANS		PROG	VARS	E-LINE

Comanda sau linia program în curs de introducere	NL	80H	Date în INPUT	NL	Spaţiu de lucru temporar
E-LINE			WORKSP		STKBOOT

Stiva calculator	Nefolosit	Stiva PROC	Stiva GOSUB	?	3EH	Caractere grafice definite de utilizator
STKBOOT	STKEND			RAMTOP		UDG P-RAMT

diverse zone de memorie. Ele nu sunt variabile BASIC şi deci nu pot fi recunoscute de calculator.

Fişierul display stochează imaginea televizorului. În loc de **PEEK** şi **POKE**, pentru imaginea display-ului se pot utiliza **SCREEN$** şi **PRINT AT** sau **PLOT** şi **POINT**.

Atributele sunt culorile, etc. pentru fiecare poziţie de caracter (se află cu instrucţiunea **ATTR**). Ele sunt stocate linie cu linie în ordinea dorită.

Buffer-ul imprimantei stochează caracterele destinate imprimantei.

Informaţiile de canal sunt necesare când se lucrează cu dispozitive de intrare-ieşire. Şi lucrul cu tastatura necesită această zonă deoarece partea de jos a ecranului funcţioneaza ca un port de intrare, în timp ce restul ecranului se comportă ca un port de ieşire.

Orice linie de comandă are forma:

2 bytes	2 bytes		00001101
n	m	t	e

unde:

1. n - este numărul liniei curente
2. m - este lungimea textului + CR
3. t - este textul liniei
4. e - este codul caracterului CR

Modul de memorare al variabilelor numerice este:

Nume	Exp	Mantisa

unde:

1. Nume - este un număr de octeţi egal cu numărul de caractere ce formează identificatorul variabilei
2. Exp - este un octet ce conţine exponentul numărului
3. Mantisa - este un grup de 4 octeţi ce conţine mantisa numărului. Bitul cel mai semnificativ al primului octet este bitul de semn.

3.17 PRODUCEREA SUNETELOR

Cuprins: BEEP

Pentru producerea sunetelor, se foloseşte instrucţiunea:

BEEP d,i

unde:

1. d - este o expresie numerică ce indică durata în secunde a sunetului respectiv
2. i - este o expresie numerică ce reprezintă înălţimea sunetului, măsurat în semitonuri relativ la DO central.

Pentru a transcrie muzica este indicat să se scrie pe marginea fiecărui spaţiu şi linie a portativului înălţimea corespunzatoare, ţinând cont de armura cheii.

Exemplu:

```
10 PRINT "Frere Gustav"
20 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
30 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
40 BEEP 1,3:BEEP 1,5:BEEP 2,7
50 BEEP 1,3:BEEP 1,5:BEEP 2,7
60 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:BEEP .5,3:
   BEEP .5,2:BEEP1,0
70 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:BEEP .5,3:
   BEEP .5,2:BEEP1,0
80 BEEP 1,0:BEEP 1,-5:BEEP 2,0
90 BEEP 1,0:BEEP 1,-5:BEEP 2,0
```

Pentru alcătuirea programului s-a procedat după cum urmează:

1. s-au adăugat mai întâi deasupra şi dedesubt câte o linie de referinţă
2. s-au numerotat liniile şi spaţiile, observând că mi bemol din armura cheii afectează nu numai mi de sus (coborându-l de la 16 la 15) cât şi mi de jos (coborându-l de la 4 la 3)

Pentru a schimba cheia partiturii, trebuie să se adune la înălţimea fiecarei note o variabilă (de exemplu "Cheie") căreia trebuie să i se atribuie valoarea adecvată înaintea execuţiei piesei.

Linia 20 a programului devine:

```
20 BEEP 1, Cheie 0:BEEP1
```

În acest exemplu variabila "Cheie" trebuie să aibă valoarea 0 pentru DO minor, 2 pentru RE minor, 12 pentru DO minor în octava superioară, etc.

Cu acest sistem este posibilă acordarea calculatorului cu un alt instrument, folosind valori zecimale pentru variabila "Cheie". De asemenea, este posibil să se execute piese cu viteze diferite. În exemplul dat "o pătrime" a fost programată să dureze o secundă. Dacă se introduce o variabilă "PATRIME" analog cu "Cheie", linia 20 devine:

```
20  BEEP patrime, cheie+0:  BEEP  patrime,
    cheie+2:BEEP patrime/2, cheie+3:BEEP patrime/2,
    cheie+2:BEEP patrime,cheie+0
```


În acest fel este posibilă execuţia aceluiaşi program în orice cheie, cu orice acordare.

Programul de mai jos:

```
FOR n=0 TO 1000: BEEP 0.5 , n: NEXT n
```

va produce note din ce în ce mai acute, pâna la limita posibilitatilor calculatorului, când acesta va tipări mesajul:

```
B   integer out of range
```

Tipărind n se obţine înalţimea notei celei mai acute care poate fi produsă. Procedeul poate fi repetat pentru notele joase.

Sunetele din gama medie sunt cele mai potrivite pentru a fi redate. Sunetele grave se aud ca nişte păcănituri. Ele pot fi prelungite pentru a deveni mai naturale, cu comanda:

```
POKE 23609, m
```

cu m=0,...,255.

3.18 UTILIZAREA CODULUI MAŞINĂ

Cuprins: USR

Calculatorul HC poate fi dotat cu un asamblor înregistrat pe casetă sau în EPROM. Introducerea programului scris în limbaj maşină (funcţie executată în general de asamblor) se face în general cu specificarea adresei de început (cel mai bine este ca aceasta adresă să se afle între zona BASIC şi zona caracterelor grafice definite de utilizator).

La pornirea unui calculator HC începutul memoriei RAM, RAMTOP se află la adresa 65366 (vezi fig. 3.2), dar se poate deplasa RAMTOP cu comanda **CLEAR** 65266 obţinându-se neutilizarea de către sistem a 100 octeţi începând cu adresa 65267 (vezi fig. 3.3).

Pentru a insera codurile obiect în memorie, se poate utiliza un program de genul:

		Grafice definite de utilizator	
	RAMTOP=65366	UDG=65367	P-RAMT=65534

Fig. 3.2.

		100 bytes liberi	Grafice definite de utilizator	
	RAMTOP=65265	65266	UDG=65367	P-RAMT=65534

Fig. 3.3..

```
10 LET a = 32500
20 READ n: POKE a,n
30 LET a = a + 1: GO TO 20
40 DATA 1,99,0,201
```

care introduce programul:

```
LD bc,99
RET
```

transpus în cod maşină ca:

1, 99, 0 (pentru LD bc,99) şi 201 (pentru RET).

Când se termină cei 4octeţi specificaţi, apare mesajul:

E Out of DATA

Rularea programului introdus în cod maşină se face cu instrucţiunea:

USR adresa de inceput

În exemplul de mai sus, cu:

PRINT USR 32500

se tipăreşte valoarea 99 din perechea de registre BC.

Adresa de revenire în BASIC se memorează cu instrucţiunea Z80 RET. În rutinele scrise în limbaj maşină nu se pot folosi registrele index IY şi IX.

Calculatorul HC are scoase în exterior magistralele de date, adrese şi de control prin intermediul unui conector de extensie.

Un program în limbaj maşină poate fi memorat ca o informaţie de tip byte; deci cu:

SAVE "nume" CODE 32500,4

se memorează programul exemplu.

Un program în limbaj de asamblare nu se poate lansa automat, odată încărcat; el poate fi însă lansat de un program în BASIC ca în exemplul:

```
10 LOAD "" CODE 32500,4
20 PRINT USR 32500
```

După aceasta se execută:

SAVE "nume" LINE

şi apoi

SAVE "nume" CODE 32500,4

Rebobinând caseta şi scriind:

LOAD "nume"

se încarcă şi se execută programul BASIC care, la rândul său va apela programul în limbaj maşină.

3.19 UTILIZAREA PORTURILOR INPUT, OUTPUT

Cuprins: IN ,OUT

Calculatorul HC dispune de 65536 adrese de memorie de tip RAM şi ROM ...nizate pe opt biţi. El poate să scrie cuvinte în memoria de tip RAM şi poate să ...ască cuvinte din memoriile de tip RAM şi ROM. Analog sunt 65536 porturi de ...PUT şi de OUTPUT. Aceste porturi sunt folosite de procesor pentru a comunica ...xteriorul. Instrucţiunile sunt:

IN adresa port

...e preia octetul citit de la acel port;

OUT adresa port, valoare

...valoarea în portul de adresă specificat. Există un ansamblu de adrese de ...care citeşte tastatura şi conectorul de casetofon. Tastatura este împărţită în ...pagini de 5 taste fiecare. Lista porturilor utilizate este:

IN 65278 citeşte semipagina CAPS SHIFT - v

Aceste adrese sunt $254 + 256*(255-2^n)$ cu n = 0,...,7

Biţii d0,..,d4 sunt asociaţi celor 5 taste din semipagina specificată. D6 este asociat conectorului de casetofon.

Portul de ieşire cu adresa 254 controlează difuzorul (D4), conectorul de casetofon (D3) şi determină culoarea chenarului (D2, D1, D0). Portul de adresă 251 controlează imprimanta în scriere şi citire; la citire verifică dacă imprimanta este gata să imprime o nouă linie şi la scriere trimite linia care trebuie să fie tipărită. Porturile de adrese 254, 247 şi 239 sunt folosite pentru echipamentele suplimentare (capitolul Alte periferice).

3.20 ÎNREGISTRAREA PE CASETĂ

Cuprins: SAVE, VERIFY, LOAD, MERGE

Calculatorul HC are posibilitatea să înregistreze programe pe bandă magnetică pe orice tip de casetofon audio.
Conectarea calculatorului la casetofon se face cu ajutorul unui cablu special.
Pentru a memora un program pe bandă, acesta trebuie să primească un nume compus din maximum 10 caractere, litere şi/sau cifre. Comanda este:

Save "nume"

Calculatorul răspunde cu mesajul:

Start tape then press any key.

La terminarea înregistrării apare mesajul:

0 OK.

Pentru verificare se reglează volumul casetofonului la nivel mediu şi se conectează cablul; se pozitionează banda în punctul în care a început înregistrarea. Comanda este:

VERIFY "nume"

În acest fel se verifică dacă programul şi variabilele înregistrate pe casetă sunt identice cu cele din memoria calculatorului. Dacă programul a fost înregistrat şi chemat corect, pe ecran apare:

Program "nume"

(În timpul căutării programului specificat, calculatorul tipăreşte numele tuturor programelor pe care le întâlneşte) şi la sfârşit mesajul:

0 OK.

În cazul unei erori de înregistrare (eroare ce apare la VERIFY) se afişează mesajul:

R Tape loading error

şi se încearcă o nouă înregistrare. Încărcarea unui program memorat pe casetă se face cu comanda:

LOAD "nume"

Această comandă şterge vechiul program (şi variabilele sale) din calculator înainte de a încărca unul nou.

LOAD ""

fără a fi urmat de un nume de program încarcă primul program găsit pe casetă.

Comanda **MERGE** încarcă un program înregistrat pe casetă în memoria calculatorului, dar spre deosebire de comanda **LOAD**, anulează din vechiul program, înaintea începerii transferului doar acele linii şi variabile cu numere, respectiv nume deja existente în programul ce urmează a fi încărcat. Dacă instrucţiunile VERIFY, **LOAD** şi **MERGE** sunt urmate de un şir vid ca nume al fişierului căutat, calculatorul va lucra asupra primului program pe care îl întâlneşte.
Este posibil să se înregistreze un program pe casetă, astfel încât atunci când este reîncărcat în memorie, el se lansează automat de la o linie specificată. Instrucţiunea este:

SAVE şir LINE număr

şi face ca programul încărcat cu **LOAD** (dar nu şi cu **MERGE**) să fie rulat automat de la linia specificată cu "număr". Dacă nu este loc suficient în memorie, programul vechi şi vechile variabile nu sunt şterse şi apare eroare:

Out of memory

În afară de programe şi variabile se mai pot memora matrici şi octeţi. Pentru memorarea unei matrici se foloseşte instrucţiunea:

SAVE şir DATA matrice()

unde:

1. şir - este numele de pe bandă al matricii
2. matrice - specifică numele matricii care va fi memorată (numerică sau şir de caractere).

Exemple:
SAVE "test" DATA b()

În acest caz se caută pe casetă o matrice cu numele "test". Când o găseşte trimite mejajul:

Number array: test

Matricea găsită este comparată cu matricea B din memorie.

LOAD "test" DATA b()

Se caută matricea pe bandă şi dacă este memorie liberă suficientă, anulează o eventuală matrice B preexistentă, şi încarcă noua matrice pe bandă denumind-o B.

MERGE nu poate fi folosit la înregistrarea matricilor pe bandă.

Memorarea tip octet este folosită pentru orice tip de dată, fără vreo referire asupra utilizării acestei date. Memorarea tip octet se face cu:

SAVE şir CODE primul octet, numărul de octeţi

Acest mod de memorare copiază o parte din memoria internă a calculatorului, aşa cum este, pe bandă. Transferul în sens invers se face cu:

LOAD şir CODE adresa de început, lungime

Când nu se specifică lungimea şirului de octeţi, calculatorul va încărca toţi octeţii înregistraţi pe casetă.

Exemplu:

Zona de memorie în care se păstrează imaginea pentru display începe la adresa 16384 şi are 6912 octeţi. Comanda:

SAVE "imagine" CODE 16384,6912

copiază imaginea de pe ecran în momentul execuţiei comenzii, pe bandă, cu numele imagine.

CODE 16384,6912 este folosită frecvent; de aceea a fost abreviată sub forma:

SCREEN$

La memorarea imaginii video nu poate fi folosită comanda **VERIFY**.

3.21 IMPRIMANTA

Cuprins: LLIST, LPRINT, COPY

Comenzile **LPRINT** şi **LLIST** sunt identice cu **PRINT** şi **LIST**, tipărind pe imprimantă, nu pe televizor.

Comanda **COPY** tipăreşte la imprimantă o copie a ecranului televizorului. **COPY** nu are efect în cazul listărilor automate (de câte ori se apasă CR).

Pentru a obţine un listing se poate folosi **LIST** urmat de **COPY** sau numai **LLIST**. Imprimanta poate fi oprită în timpul unei tipăriri acţionând **BREAK**.

3.22 VARIABILE DE SISTEM

Octeţii din memorie de la adresa 23552 la adresa 23733 sunt rezervaţi pentru operaţii specifice ale sistemului. Ei pot fi citiţi pentru a afla diferite lucruri despre sistem, iar câţiva din ei pot fi şi modificaţi. Aceşti octeţi se numesc variabile de sistem, şi au câte un nume, dar nu trebuie confundaţi cu variabilele utilizate de BASIC. În cazul variabilelor formate din mai mulţi octeţi, primul va fi octetul cel mai puţin semnificativ. Variabilele de sistem sunt date în lista de mai jos. Abrevierile din coloana 1 au următoarea semnificaţie:

X această variabilă nu poate fi modificată deoarece sistemul va funcţiona eronat
N modificarea acestei variabile nu are un efect asupra funcţionării normale a sistemului
n numărul de octeţi din variabilă

Tip	Adresa	Nume	Conţinut
N8	23552	KSTATE	Folosită în citirea tastaturii
N1	23560	LAST K	Reţine ultima tastă apăsată
1	23561	REPDEL	Durata (în 1/50 sec) cât trebuie ţinută apăsată o tastă pentru a se repeta
1	23562	REPPER	Timpul (în 1/50 sec) după care se repetă o tastă apăsată
N2	23563	DEFADD	Adresa argumentelor funcţiilor definite de utilizator
N1	23565	K DATA	Al doilea octet pentru controlul culorii introdus de la tastatură
N2	23566	TVDATA	Controlul culorii, al lui AT şi TAB pentru TV
X38	23568	STRMS	Adresa canalului ataşat căii
2	23606	CHARS	Adresa generatorului de caractere minus 256
1	23608	RASP	Durata sunetului de eroare
1	23609	PIP	Durata sunetului la apăsarea unei taste
1	23610	ERR NR	Codul de mesaj minus 1
X1	23611	FLAGS	Diferiţi indicatori de control ai sistemului BASIC
X1	23612	TVFLAG	Indicatori asociaţi cu televizorul
X2	23613	ERR SP	Adresa elementului din stiva maşinii utilizat ca adresă de întoarcere în caz de eroare
N2	23615	LIST SP	Adresa de întoarcere la listările automate
N1	23617	MODE	Specifică cursorul (K,L,C,E,G)
2	23618	NEWPPC	Linia la care se sare
1	23620	NSPPC	Numărul instrucţiunii în linie la care se sare
2	23621	PPC	Numărul liniei pentru instrucţiunea în execuţie
1	23623	SUBPPC	Numărul instrucţiunii din linie în execuţie
1	23624	BORDCR	Culoarea border-ului
2	23625	E PPC	Numărul liniei curente
X2	23627	VARS	Adresa variabilelor
N2	23629	DEST	Adresa variabilelor asignate
X2	23631	CHANS	Adresa datelor de canal

X2	23633	CURCHL	Adresa informației curente folosită pentru intrare sau ieșire
X2	23635	PROG	Adresa programului BASIC
X2	23637	NXTLIN	Adresa următoarei linii din program
X2	23639	DATADD	Adresa ultimului element din lista DATA
X2	23641	E LINE	Adresa comenzii introduse
2	23643	K CUR	Adresa cursorului
X2	23645	CH ADD	Adresa următorului caracter care urmează să fie interpretat
2	23647	XPTR	Adresa caracterului după semnul întrebării
X2	23649	WORKSP	Adresa spațiului de lucru temporar
X2	23651	STKBOT	Adresa inferioară a stivei calculator
X2	23653	STKEND	Adresa de început a spațiului liber
N1	23655	BREG	Registrul B al calculatorului
N2	23656	MEM	Adresa spațiului folosit pentru memoria calculatorului
1	23658	FLAGS2	Alți indicatori
X1	23659	DF SZ	Numărul liniilor din partea de jos a ecranului
2	23660	S TOP	Numărul liniei de sus a programului la listarea automată
2	23662	OLDPPC	Numărul liniei la care sare CONTINUE
1	23664	OSPCC	Numărul din linie la care sare CONTINUE
N1	23665	FLAGX	Diverși indicatori
N2	23666	STRLEN	Lungimea asignată șirului
N2	23668	T ADDR	Adresa următorului element din tabela sintaxă
2	23670	SEED	Variabila pentru RND
3	23672	FRAMES	Contorul de cadre
2	23675	UDG	Adresa primului grafic definit de utilizator
1	23677	COORDS	Coordonata x a ultimului punct plot-at
1	23678		Coordonata y a ultimului punct plot-at
1	23679	P POSN	Numărul poziției de scriere pe ecran
1	23680	PR CC	Octetul mai puțin semnificativ al adresei pentru noua poziție la care se imprimă prin LPRINT
1	23681		Nefolosit
2	23682	ECHO E	Numărul coloanei și al liniei
2	23684	DF CC	Adresa de afișare pe ecran prin PRINT
2	23686	DFCCL	Același lucru pentru partea de jos a ecranului
X1	23688	S POSN	Numărul coloanei pentru PRINT
X1	23689		Numărul liniei pentru PRINT
X2	23690	SPONSNL	Ca S POSN pentru partea de jos a ecranului
1	23692	SCR CT	Numără defilările pe ecran
1	23693	ATTR P	Culoarea curentă
1	23694	MASK P	Folosit pentru culori transparente
N1	23695	ATTR T	Culori temporare
N1	23696	MASK T	Ca MASK P dar temporar
1	23697	PFLAG	Alți indicatori

N30	23696	MEMBOT	Arie memorie calculator
2	23728		Nefolosit
2	23730	RAMTOP	Adresa ultimului octet din aria sistemului BASIC
2	23732	P-RAMT	Adresa ultimului octet de RAM

3.23 CANALE I/O ȘI CĂI

Cuprins: INPUT#, PRINT#, OPEN#, CLOSE#, LIST#, INKEY$#

Pentru fiecare echipament periferic sau port I/O este asignată o linie de comunicație numită canal. Fiecărui canal existent i se poate asocia o parte componentă software numită cale. Pentru a transmite informații pe un canal oarecare este suficient să transmitem informațiile pe calea asignată acestui canal.

Exemplu:

INPUT# s; 'lista variabile'

citește date de la portul asignat căii s și le asociază variabilelor din lista de variabile. Similar

PRINT# s; 'lista variabile'

trimite date către portul asociat căii s.

Asignarea unei căi la un echipament I/O se face cu instrucțiunea **OPEN# s,c** unde:

s este numărul căii
c este un șir care specifică canalul

Instrucțiunea **OPEN#** realizează și inițializarea echipamentului I/O. Unui canal i se pot asocia mai multe căi.
În configurația de bază calculatorul HC recunoaște trei canale:

canalul K - claviatura
canalul S - ecran
canalul P - imprimanta

Canalele S și P sunt canale pe care se poate doar scrie la echipamentul I/O.

Exemplu:

```
10 OPEN# 5,"K"
20 PRINT# 5,"HC"
30 GO TO 20
```

trimite date la ieşirea căii 5 care este asociată prin instrucţiunea **OPEN#** părţii de jos a ecranului.

Pentru a anula asignarea căii s la un canal se foloseşte instrucţiunea **CLOSE#** s. După instrucţiunea **CLOSE#** calea s poate fi asociată altui canal.

La iniţializarea sistemului se deschid automat căile 0-3, cu următoarea asignare:

calea 0 - canalul K
calea 1 - canalul K
calea 2 - canalul S
calea 3 - canalul P

Instrucţiunea **LIST#** **s,n** listează programul începând cu linia n pe calea s. Comanda **INKEY$#** s citeşte un octet de pe calea s.

3.24 ALTE ECHIPAMENTE

Reţea

Poate fi folosită o periferie de tip reţea pentru conectarea mai multor calculatoare HC între ele.

Interfaţa serială

Interfaţa standard RS-232 permite conectarea unui HC cu alt calculator sau alte periferice înzestrate cu această interfaţă. Utilizarea se realizează folosind cuvintele cheie **OPEN#, CLOSE#, MOVE, ERASE, CAT** şi **FORMAT**.

Interfaţa disc flexibil

Interfaţa de disc flexibil permite cuplarea a unu sau două minidrive-uri. Acestea au avantajul unei operaţiuni de încărcare-salvare mult mai sigură şi mai rapidă în comparaţie cu caseta.

Interfaţa de creion optic şi Kempston

Această interfaţă dă posibilitatea utilizatorului să cupleze un Joystick tip KEMPSTON, pentru jocuri sau aplicaţii practice şi cuplarea unui creion optic folosit pentru desenat.

Capitolul 4. INTERFATA 1

4.1. PREZENTARE GENERALĂ

Calculatorul HC-2000 înglobează pe placa sa de bază încă trei interfeţe: cu discul flexibil (sau floppy), cu o linie serială standard CCITT V24 (RS-232C) şi o interfaţă mai puţin obişnuită care permite cuplarea mai multor HC-uri 2000 (sau HC 85/90/91/92 dotate cu "Interfaţa 1" - IF1) printr-o singură pereche de fire torsadate, conexiune denumită reţea.

Din punct de vedere constructiv, interfeţele calculatorului HC- 2000 realizează exact aceleaşi funcţii ca placa de extensie "Interfaţa 1" (IF1) ataşată calculatoarelor anterioare ale gamei HC (HC85, HC90, HC91) dar care se livra separat de calculator.

Interfaţa de floppy permite cuplarea minidiscului intern de 3.5 ţoli, 80 piste, 720K şi opţional a unui minidisc exterior de 3.5 ţoli sau de 5.25 ţoli, 40 piste, 360K, spaţiu care poate fi utilizat pentru a memora până la 64 de fişiere distincte.

Rata medie de transfer a discului este de 25 până la 30 de ori mai mare decât a interfeţei standard de casetă magnetică. Dacă mai adaugăm la aceasta şi accesul aleator la informaţii (timpul maxim de acces la un sector de disc este de circa 1.7 secunde), este imposibil să nu remarcăm avantajele majore faţă de interfaţa de casetă.

Interfaţa serială rezolvă în principal problema cuplării unei imprimante seriale la HC, dar poate fi folosită şi pentru a transfera date cu orice alt tip de calculator dotat cu interfaţa RS-232C, folosind bineînţeles programe speciale de transfer.

Interfaţa de reţea oferă o soluţie pentru una din aplicaţiile posibile ale calculatoarelor HC: învăţământul. Cuplate într-o reţea de până la 64 de sisteme, reţeua HC-urilor dintr-o sală de laborator informatic poate uşura atât sarcina profesorului, cât şi sarcina elevilor.

Viteza de transfer a informaţiei prin reţea este de 80 kiloocteţi pe secundă. Transferurile de date se fac în blocuri cu lungime variabilă (maxim 255 octeţi), însoţite de blocuri de control care specifică adrese sursa/destinaţie, număr bloc, etc. Protocolul este suficient de cuprinzător pentru a permite schimbul simultan de mesaje între oricâte noduri ale reţelei folosind numai două fire torsadate pentru a lega nodurile între ele.

Din punct de vedere al programării, interfeţele se integrează în sistemul BASIC al calculatorului HC, oferind fie noi instrucţiuni, fie extensii ale instrucţiunilor existente. Extinderea limbajului BASIC se face fără nici o modificare a plăcii de bază, interfeţele interceptând prin hardware rutina de eroare din placa de bază.

Cele trei interfeţe oferă pe lângă facilităţi de încărcare/salvare programe şi date, comenzi pentru manipularea de fişiere, ceea ce oferă o nouă dimensiune în stocarea şi regăsirea datelor folosind programe scrise în BASIC.

Demn de subliniat este faptul că HC-2000 poate fi actionată şi în CP/M. CP/M-ul este un sistem de operare profesional pe care îl găsiţi instalat de regulă pe orice calculator realizat în jurul microprocesorului INTEL 8080 sau ZILOG Z80.

Lucrând sub CP/M calculatorul HC-2000 se apropie ca performanţe de calculatoarele "serioase" (M-118, CUB-Z, etc.).

4.2 UNITATEA DE DISC FLEXIBIL

Interfaţa de disc flexibil este realizată cu un controler de disc flexibil 8272 (FDC) care asigură semnalele de comandă pentru interfaţarea calculatorului cu 2 unităţi de disc. Acest circuit poate lucra fie în format simplă densitate (FM IBM) sau în format dublă densitate (MFM), inclusiv dublă faţă.

Aşa cum am arătat în paragraful precedent, HC-2000 este dotat cu o unitate internă de disc flexibil de 3.5 ţoli, dublă faţă, 80 piste, 720K.

Atenţie! Nu porniţi sau opriţi calculatorul având discurile introduse în unitatea de disc. Informatiile de pe disc pot fi distruse.

Nu recomandăm folosirea dischetelor de 3.5 toli HD (1.44M).

Pentru a lucra cu discul trebuiesc ştiute următoarele lucruri:

-Inserarea discului:

Discul se introduce în unitatea de disc cu eticheta în sus (deci cu sistemul de rotire a dischetei în jos) şi cu uşiţa metalică de protecţie a dischetei înainte.

Dacă decupajul din stânga este descoperit, discul este protejat la scriere. Discurile HD (nerecomandate) prezintă un al doilea decupaj, în partea dreaptă. Acesta nu influenţeaza în nici un fel funcţionarea minidiscului, deci ignoraţi-l!

-Manipularea discului:

a. Nu deschideţi niciodată usiţa metalică de protecţie a dischetei!
b. Nu apropiaţi magneţi de disc.
c. Introduceţi discul în plic după scoaterea din unitate.
g. Protejaţi discul de lichide , praf şi scrum de ţigară.
h. Păstraţi discul la o temperatură între 10^{o} C şi 55^{o} C şi o umiditate relativă între 8% şi 80%.

4.3 PRIMELE OPERAŢII CU MINIDISCUL

În cele ce urmează ne vom referi la minidiscul intern al HC- 2000 denumindu-l "Minidrive-ul 1" sau mai simplu "Drive-ul nr. 1".

Auto-run

Poate sunteţi curios să aflaţi ce programe vă aşteaptă pe discul de demonstraţie. Pentru aceasta, inseraţi floppy discul în Minidrive (sau dacă aveţi două Minidrive-uri, în Minidrive-ul 1), şi introduceţi:

NEW

urmat de:

RUN (şi RETURN)

Aceste comenzi vor declanşa încarcărea automată şi rularea primului program de pe floppy. După ce aţi terminat de privit acest program, citiţi mai departe.

Catalogul

Pentru a afla ce alte programe se găsesc pe floppy-ul de demonstraţie, introduceţi instrucţiunea CATalog:

```
CAT 1
    |
    +--- 1 identifică numărul Minidrive-ului pe care îl folosiţi
```

În aproximativ 3 secunde pe ecranul televizorului se va afişa:

- un catalog al tuturor numelor fişierelor memorate pe floppy;
- spaţiul rămas disponibil pe floppy (în kilocteţi).

Încărcarea programelor

Următorul lucru de făcut este încărcarea programului pe care vreţi să-l executaţi în continuare. Pentru asta alegeţi mai întâi un program, apoi introduceţi:

```
LOAD *"d";1;"nume"
    |  |     |
    |  |     +- aici introduceţi numele programului pe care l-aţi ales.
    |  |
    |  +-- "d";1; identifică ce Minidrive folosiţi.
    |
    +---- steluţa comunică calculatorului că folosiţi un Minidrive,
          şi nu interfaţa obişnuită de casetă.
```

După o scurtă pauză, ecranul va afişa mesajul OK (dar fără numele programului). Puteţi acum lansa programul în execuţie (cu RUN).

4.4. UTILIZARE MINIDISC PENTRU PROGRAME

Salvare, verificare, încărcare şi comasare programe

În manualul de utilizare HC-2000 aţi găsit instrucţiunea SAVE, care salvează programe pe casetă. Salvarea programelor pe floppy este la fel de simplă. Pentru exemplificare va fi folosit programul de mai jos, denumit Patrate. El tipăreşte numerele de la 1 la 10 împreună cu pătratele lor.

```
10 REM Patrate
20 FOR n = 1 TO 10
30 PRINT n,n*n
40 NEXT n
```

Pentru a salva acest program pe casetă, aţi fi introdus:

SAVE "Patrate"

Pentru a-l salva pe floppy-ul din Minidrive-ul 1, introduceţi:

```
SAVE *"d";1;"Patrate"
```

După câteva secunde în care marginea ecranului va clipi, programul va fi salvat. (Numele programelor memorate pe floppy pot avea o lungime maximă de 11 caractere).

Aşa cum probabil v-aţi imaginat deja, puteţi verifica corecta înregistrare a programului pe floppy introducând:

```
VERIFY *"d";1;"Patrate"
```

Ecranul va afişa mesajul OK.
Puteţi încărca acum programul Patrate introducând:

```
NEW
```

urmat de:

```
LOAD *"d";1;"Patrate"
```

În continuare, pentru a face ca programul să se lanseze automat, încercaţi să introduceţi:

```
SAVE *"d";1;"Patrate2" LINE 10
```

apoi:

```
NEW
```

şi apoi:

```
LOAD *"d";1;"Patrate2"
```

Minidrive-ul poate fi folosit şi pentru a comasa programele.
Introduceţi:

```
NEW
```

urmat de:

```
100 REM alte Patrate
110 FOR n = 11 TO 20
120 PRINT n,n*n
130 NEXT n
```

şi acum introduceţi:


```
MERGE *"d";1;"Patrate"
```

şi programul Patrate va fi adăugat la listing.

Pe scurt, aşa cum v-aţi dat deja seama, sintaxa folosită pentru obişnuita interfaţă de casetă (explicată în secţiunea "Memorare pe bandă" din manualul de programare BASIC) se aplică şi la Minidisc.

Ştergerea programelor

Să presupunem că aţi terminat de lucrat cu programul Patrate. Pentru a-l şterge, introduceţi:

```
ERASE "d";1;"Patrate"
```

(Ca înainte, "d";1 indică ce Minidrive folosiţi).
În timpul execuţiei intrucţiunii ERASE, marginea ecranului va clipi.

Formatarea discurilor

Înainte de prima utilizare a unui disc floppy, inseraţi-l într-un Minidrive (de exemplu Minidrive-ul 1) şi introduceţi:

```
FORMAT "d";1
```

"d";1 identifică Minidrive-ul pe care îl folosiţi (în acest caz Minidrive-ul 1).

Formatarea unui disc durează aproximativ treizeci de secunde. În timpul acesta, marginea ecranului se va schimba la început, şi va reveni puţin înainte de afişarea mesajului OK. Procesul de formatare constă din iniţializarea fiecărei piste de pe floppy, prin scrierea câmpurilor de identificare şi date corespunzătoare fiecărui sector. După formatarea unei piste, fiecare sector în parte este citit, verificând suma de control. Mesajul OK apare numai dacă toate sectoarele au putut fi citite corect (nu se acceptă discuri cu sectoare eronate).
Formatarea unui floppy nu trebuie repetată niciodată, şi pentru că prin formatarea unui disc se pierde orice a fost înregistrat pe el.
Apăsaţi acum:

```
CAT 1
    |
    +--- 1 identifică numărul minidrive-ului pe care îl folosiţi
```

După câteva secunde, în care timp marginea ecranului va clipi, va apare mesajul de eroare:

```
File not found
```

care semnifică faptul că floppy-ul nu conţine nici un program.

Capacitatea unui disc este de 720 kiloocteţi.

Instalarea facilităţii de auto-run

Puţin mai înainte aţi folosit facilitatea de auto-run pentru discul de demonstraţie. Dacă aveţi un program pe care îl folosiţi adesea, va puteţi stabili propria facilitate de auto-run, astfel încât să nu mai introduceţi instrucţiunile LOAD şi RUN. Acestea sunt regulile de urmat:

- programul trebuie să aibă numele run;
- floppy-ul trebuie folosit în Minidrive-ul 1;
- facilitatea trebuie folosită fie imediat după punerea sub tensiune, sau imediat dupa comanda NEW.

Astfel, introduceţi programul respectiv, urmat de comanda:

```
SAVE *"d";1;"run" LINE numar
          |           |
          |           +--- introduceţi aici numărul liniei de start
          |
          +- numele run trebuie  introdus literă cu literă. Nu
apăsaţi
                 tasta RUN!
```

Acum introduceţi:

```
NEW
```

urmat de:

```
RUN
|
+--- Tasta RUN, şi nu numele programului.
```

4.5. DATE, CANALE ŞI CĂI

Precum ştiţi, un program este un set de operaţii care se execută atunci când apăsaţi RUN. Datele, pe de altă parte, sunt orice colecţie de litere, numere sau simboluri cu care poate lucra un program. Exemple sunt numerele de la 1 la 10 şi pătratele lor.

Datele pot fi trimise, sau receptionaţe, către/de la diferite părţi ale unui sistem de calcul. Aceste părţi sunt denumite 'canale'. Canalele către care se pot trimite date sunt:

- ecranul televizorului
- un fişier pe floppy

- un alt calculator HC-2000, dacă amândouă calculatoarele sunt cuplate printr-o reţea.
- interfaţa RS232 şi de acolo, de exemplu, la un modem sau o imprimantă.

Canalele de la care se pot primi date sunt:

- claviatura
- un fişier pe floppy
- un alt HC-2000, dacă amândouă calculatoarele sunt cuplate printr-o reţea.
- interfaţa RS232, adică un modem sau un terminal.

Nodurile de comunicaţie dintre programul BASIC şi canale sunt denumite căi. În sistemul HC-2000, numărul acestor căi este fixat la 16. Ele sunt numerotate de la 0 la 15, iar numerele de cale sunt intotdeauna precedate de semnul #.
Patru dintre aceste căi sunt deja cuplate la următoarele canale:

calea #0 trimite date catre partea de jos a ecranului TV şi
calea #1 primeşte date de la claviatură;
calea #2 trimite date către partea de sus a ecranului TV, dar nu poate primi date;
calea #3 trimite date către imprimantă, dar nu poate primi date.

Orice intrucţiune care execută un transfer de intrare/iesire foloseşte una din aceste căi în mod implicit. De exemplu, instrucţiunea PRINT foloseşte calea #2, iar instrucţiunea LPRINT foloseşte calea #3. Astfel, dacă introduceţi:

```
PRINT "Acesta este un calculator HC-2000"
```

folosiţi de fapt o prescurtare a instrucţiunii:

```
PRINT #2;"Acesta este un calculator HC-2000"
```

Verificaţi prin introducerea celor două forme.
Puteţi, totuşi, să faceţi fiecare instrucţiune să folosească o altă cale prin introducerea semnului # urmat de un număr de cale. Încercaţi să introduceţi:

```
LPRINT #2;"Acesta este un calculator HC-2000"
```

în loc să fie trimis la imprimantă, acest mesaj apare pe ecranul TV.

Dar în loc să folosiţi căile prestabilite, puteţi crea unele proprii. Căile #4 până la #15 sunt rezervate pentru acest scop; şi există diferite 'specificatoare de canale' care indică perifericul dorit. Câteva exemple sunt:

"K" pentru claviatura
"S" pentru ecran
"P" pentru imprimantă

(altele vor fi introduse mai târziu).

Remarcaţi faptul că K, S şi P sunt toate canale prestabilite. Ele solicită utilizarea virgulelor (,) drept separatori în instrucţiunile OPEN #. Dar cu alte canale puteţi folosi fie virgule fie punct-virgulă (;).

Pentru a crea o cale proprie folosiţi instrucţiunea OPEN #. De exemplu introduceţi:

```
10 OPEN #4,"S"
```

Astfel deschideţi calea #4 şi îl cuplaţi la canalul "S". Acum introduceţi:

```
20 PRINT #4;"Acesta este un calculator HC-2000"
```

şi din nou linia va apare pe ecran.

(Nu se recomandă deschiderea căilor 0, 1 sau 2, pentru că rezultatele acestor operaţii pot fi imprevizibile).

4.6. FIŞIERE DE DATE PE DISC

Deschiderea unui fişier de date

Memorarea informaţiilor pe floppy se face în fişiere. Fiecare fişier primeşte la creare un nume, pentru a putea fi regăsit mai târziu. Instrucţiunea care deschide şi denumeşte un fişier de date are întotdeauna aceeaşi formă. De exemplu tastaţi instrucţiunea:

```
OPEN #4;"d";1;"Numere"
   |   |    |
   |   |    +- "Numere" este numele fişierului. Acesta poate
   |   |       fi orice şir de caractere de lungime max. 11
   |   |
   |   +-------- "d";1 identifică Minidiscul pe care îl folosiţi
   |
   +------------ numărul de cale poate fi orice
                 număr între 0 şi 15
```

Această instrucţiune face două lucruri distincte:

- stabileşte un canal cuplat cu fişierul: "d";1;"Numere"
- ataşează acest nou canal la calea #4.

Operaţia va dura câteva secunde, în care timp calculatorul va căuta pe floppy un fişier cu numele "Numere". Pentru că nu există fişierul "Numere", deschide canalul pentru scriere. (Dacă ar fi găsit un fişier cu numele "Numere", l-ar fi deschis pentru citire).

Introducerea datelor

Odată ce aţi deschis un fişier, puteţi introduce date. Să presupunem că vreţi să memoraţi numerele de la 1 la 10 împreună cu pătratele lor. Introduceţi şi rulaţi programul următor:

```
10 FOR n = 1 TO 10
20 PRINT #4;n'n*n
30 NEXT n
```

S-ar putea să credeţi că toate numerele au fost deja memorate pe floppy. Dar de fapt calculatorul nu transferă în mod automat datele pe floppy decât după ce s-a acumulat o anumită cantitate de informaţii, pe care o transferă dintr-odată. Acest procedeu se numeşte 'blocarea' datelor. Un bloc de date pe floppy are lungimea de 256 de octeţi (sau caractere).

Pentru a memora pe floppy datele introduse trebuie să închideţi fişierul. Până nu faceţi acest lucru, nu veţi putea să citiţi din fişier.

Inchiderea unui fişier

Închiderea unui fişier asigură memorarea definitivă a datelor pe floppy. Închide de asemenea canalul (în cazul nostru "d";1;"Numere") şi detaşează calea (în cazul nostru #4) de la orice canal. Pentru a închide un fişier trebuie doar să închideţi calea asociată:

```
CLOSE #4
```

Marginea ecranului va clipi pentru a arăta că se înregistrează ceva pe floppy.

(Remarcaţi faptul că, la fel ca la instrucţiunea OPEN, instrucţiunea CLOSE este urmată în mod automat de #).

Căile #0, #1, #2, #3 rămân întotdeauna ataşate unui canal, chiar dacă se execută o instrucţiune CLOSE specifică. Dacă încercaţi să închideţi una din aceste căi, căile #0 şi #1 se vor ataşa automat la canalul K; calea #2 la canalul S; iar calea #3 la canalul P.

Citirea datelor dintr-un fişier

Pentru a citi datele din fişierul "Numere" rulaţi următorul program:

```
10 OPEN #4;"d";1;"Numere"
20 FOR b = 1 TO 10
30 INPUT #4;m;n
40 PRINT "Patratul lui ";m;" este ";n
50 NEXT b          |    |   |
60 CLOSE #4        +--  +-- +--lasaţi spaţii !
RUN
```

Pentru ca fişierul "Numere" există deja pe floppy, canalul "d";1;"Numere" este

deschis pentru intrare, şi orice încercare de a scrie date ar fi generat o eroare.

Se poate de asemenea folosi funcţia INKEY$ pentru a citi date dintr-un fişier (întoarce întotdeauna următorul caracter din fişier). Încercaţi programul următor:

```
10 OPEN #11;"d";1;"listing"
20 LIST #11
30 CLOSE #11
40 OPEN #12;"d";1;"listing"
50 PRINT INKEY$#12;
60 GO TO 50
```

Acest program se va termina cu un mesaj de sfârşit de fişier, adică End of file.

Observaţii asupra lui PRINT şi INPUT

Pentru că instrucţiunile PRINT şi INPUT au fost concepute în principal pentru utilizarea cu ecranul şi claviatura, trebuie să fiţi atenţi la folosirea lor cu fişiere.

'separatori'

Instrucţiunea PRINT are trei forme de separatori:

- semnul ; (punct-virgulă) nu tipăreşte nimic,
- semnul , (virgulă) vă aduce la începutul următoarei jumătăţi de linie,
- semnul ' (apostrof) sare la linie nouă (codul RETURN).

Instrucţiunea INPUT aşteaptă întotdeauna să introduceţi RETURN după un număr sau un şir. Astfel, de fiecare dată când tipăriţi într-un fişier din care vreţi să citiţi mai târziu cu INPUT, trebuie fie să:

- tipăriţi fiecare element separat, adică

```
10 PRINT #4;2
20 PRINT #4;3
```

sau

- separaţi elementele cu apostrof, adică

```
10 PRINT #4;2'3
```

De asemenea, în instrucţiunile INPUT, trebuie să folosiţi cu atenţie separatorii. Aşa cum ştiţi, INPUT poate tipări în partea de jos a ecranului orice se poate pune într-o instrucţiune PRINT. Dar dacă citiţi cu INPUT dintr-un fişier, fişierul se deschide numai pentru citire. Aşa încât, dacă includeţi orice s-ar fi tipărit la utilizarea ecranului, veţi obţine mesajul de eroare Writing to a 'read' file (Scriere într-un fişier de citire). Aceasta înseamnă că elementele dintr-o instrucţiune INPUT trebuie separate numai prin punct- virgulă, adică

```
10 INPUT #4;a;b
```


Atenţie de asemenea la citirea cu INPUT a şirurilor de caractere care conţin " (ghilimele), pentru că INPUT va interpreta ghilimelele drept sfârşit de şir. Metoda de a evita acest lucru este de a înlocui, de exemplu:

```
10 INPUT #4;a$
```

cu

```
10 INPUT #4; LINE a$
```

Schimbarea căilor

Instrucţiunile PRINT pot conţine informaţii pentru mai multe căi la un moment dat. Programul următor va tipări "unu" pe ecran; "doi" într-un fisier pe floppy denumit "cifre"; "trei" către staţia 1 pe reţea (vezi capitolul următor); şi "patru" în următoarea linie din ecran.

```
10 OPEN #4;"d";1;"cifre"
20 OPEN #5;"n";1
30 PRINT "unu";#4;"doi";#5;"trei"'#2;"patru"
40 CLOSE #4
50 CLOSE #5
```

'schimbarea culorilor'

După ce aţi folosit un canal diferit de ecran, se poate ca instrucţiunile PAPER şi INK să nu aibă nici un efect. Pentru a evita acest lucru, introduceţi:

```
PRINT;
```

înainte de a schimba PAPER sau INK.

Afişarea catalogului de fişiere

Pe măsură ce se înregistrează fişiere pe floppy, acestea sunt introduse în mod automat în catalog. Astfel, pentru a afla ce fişiere sunt înregistrate pe floppy, este suficient să inseraţi floppy-ul într-un Minidrive şi să introduceţi instrucţiunea CATalog. De exemplu, introduceţi:

```
CAT 1
```

Ecranul televizorului va afişa:

- numele fişierelor
- spaţiul disponibil rămas pe floppy (în kilooteţi)

Puteţi să transferaţi ieşirea unui CAT către o cale introducând:

```
CAT #numar;numar
     |      |
     |      +----- număr Minidrive
     |
     +---------- număr cale
```

Aceasta vă permite să trimiteți catalogul către o imprimantă, sau către un fişier, astfel încât să poată fi folosit de un program.

Protejarea unui fişier

Dacă doriți ca un nume să nu apară în catalog, îl puteți proteja dându-i un nume care are în poziția 10 codul caracterului dorit plus 128. Introduceți acest program:

```
10 OPEN #4,"d";1;"Rezultate" + CHR$ (128 + CODE " ")
20 FOR n = 1 TO 15
30 PRINT #4;n'n*n
40 NEXT n
50 CLOSE #4
```

Acum introduceți:

```
CAT 1
```

Numele fişierului nu va apare. Astfel că, de fiecare dată când generați un fişier protejat, amintiți-vă să-i notați numele undeva, pentru cazul în care îi uitați numele!

Extinderea unui fişier

Să presupunem că vreți să extindeți fişierul "Numere" pentru a include pătratele numerelor de la 1 la 20 în loc de numai 1 la 10. Un fişier nu poate fi redeschis pentru scriere, astfel că trebuie să:

- creați o nouă versiune cu alt nume;
- transferați vechiul fişier în noua versiune;
- adăugați noile date
- inchideți vechiul fişier.

Iată cum se poate face aceasta. Mai întâi rulați acest program:

```
10 OPEN #4;"d";1;"Numere": REM pentru citire
20 OPEN #5;"d";1;"Numere 1": REM pentru scriere
30 FOR f = 1 TO 10
40 INPUT #4;m;n
50 PRINT #5;m'n
60 NEXT f
70 FOR n = 11 TO 20
80 PRINT #5;n'n*n
```



```
90 NEXT n
100 CLOSE #4; CLOSE #5
```

Pentru a verifica existența a două fişiere, "Numere" şi "Numere 1", introduceți:

```
CAT 1
```

Apoi, ca să ştergeți vechiul fişier, introduceți:

```
ERASE "d";1;"Numere"
```

Pentru a verifica ştergerea, introduceți:

```
CAT 1
```

Numele fişierului "Numere" a dispărut din catalog, iar noul fişier, "Numere 1" conține acum numerele de la 1 la 20.

Capitolul 5. REŢEAUA LOCALĂ

5.1. CONFIGURAREA UNEI REŢELE

Reţeaua locală permite utilizatorului şi prietenilor lui să schimbe între ei programe şi date. Aceasta înseamnă că numai unul dintre voi trebuie să introducă un program. O reţea este foarte utilă şi dacă numai unul dintre voi are un Minidisc.

Folosind cablul furnizat odată cu interfaţa, puteţi lega de la două până la 64 de calculatoare HC-2000.

Configuraţia reţelei nu trebuie să fie în nici un caz o buclă închisă: calculatoarele de la capetele reţelei nu trebuie să fie conectate între ele. Fiecare capăt de reţea trebuie să aibă un conector neocupat.

NU PORNIŢI ŞI NU OPRIŢI NICIODATĂ UN HC CARE ESTE CUPLAT LA REŢEA ÎN TIMP CE ARE LOC UN TRANSFER DE DATE PE REŢEA.

Totuşi puteţi avea un HC-2000 oprit pe reţea; puteţi de asemenea să porniţi sau să opriţi HC-2000-uri care sunt pe reţea, cu condiţia să nu se facă transferuri pe reţea în acel moment.

Dupa ce aţi stabilit o reţea, fiecare calculator (sau staţie) trebuie să primească un număr de identificare diferit. Mai întâi stabiliţi împreună cu prietenii dumneavoastră, care va fi numărul fiecărei staţii, după care fiecare dintre dumneavoastră trebuie să introducă:

```
FORMAT "n";numar
              |
              +------ introduceţi numărul de staţie pe care l-aţi ales
```

Dacă reţeaua este formată numai din două calculatoare, amândouă pot folosi acelaşi număr de staţie. Şi pentru că amândouă calculatoarele devin staţia 1 în mod automat la punerea sub tensiune, utilizarea instrucţiunii FORMAT nu mai este necesară.

5.2. PROGRAMELE ŞI REŢEAUA

Să presupunem că aţi cuplat două calculatoare într-o reţea, cu numerele de staţie 1 respectiv 2.

Să presupunem că vreţi să trimiteţi către staţia 2 următorul program:

```
10 REM patrate
20 FOR n = 1 TO 10
30 PRINT n,SQR n
40 NEXT n
```

Introduceţi programul urmat de:

```
SAVE *"n";2
```


(Remarcaţi că reţeaua nu foloseşte nume pentru programe.)
Între timp la staţia 2 trebuie introdus:

```
FORMAT "n";2
```

urmat de:

```
LOAD *"n";1
```

Staţia 2 va avea acum o copie a programului. Remarcaţi cum marginea ecranului rămâne neagră în timp ce calculatorul aşteaptă să salveze sau să încarce programul prin reţea. Staţia 1 nu va trimite până când staţia 2 nu este gata, iar staţia 2 va aştepta până când se emite ceva. Încercaţi să introduceţi linia cu SAVE înainte ca la staţia 2 să introduceţi LOAD şi vice versa.

Pentru a verifica transmisia corectă a programului, la staţia 2 trebuie introdus:

```
VERIFY *"n";1
```

în timp ce la staţia 1 se repetă transmisia programului introducând:

```
SAVE *"n";2
```

SAVE este de fapt singura instrucţiune care transmite programe în reţea. Instrucţiunile LOAD, VERIFY şi MERGE sunt toate metode de a recepţiona programele.

Jocul de reţea este un bun exemplu pentru utilizarea programelor în reţea.

5.3. FIŞIERE DE DATE ÎN REŢEA

Să presupunem că doriţi să transmiteţi acum date către staţia 2. Instrucţiunea OPEN #4;"n";2 deschide un canal către staţia 2 pe reţea şi ataşează calea #4 la el, astfel încât dacă scrieţi prin calea #4, mesajul va fi pus pe reţea împreună cu o notiţă care indică sursa mesajului.

Dacă aţi fi introdus INPUT #4;m$ calculatorul dumneavoastră ar fi aşteptat informaţii adresate staţiei 1 de la staţia 2.

Acum introduceţi acest program:

```
10 OPEN #4;"n";2: REM pentru iesire
20 INPUT a$: PRINT #4;a$
70 GO TO 10
```

Apoi introduceţi:

```
SAVE *"n";2
```

Acum introduceţi la staţia 2:

```
FORMAT "n";2
LOAD *"n";1
```

Introduceţi la staţia 1 RUN, iar la staţia 2 editaţi liniile 10 şi 40 pentru a se referi la staţia 1 şi nu la staţia 2. Apoi se introduce la staţia 2:

```
GO TO 40
```

Sunteţi acum gata să începeţi o conversaţie la staţia 1 şi nu la staţia 2. Apoi se introduce la staţia 2:

```
GO TO 40
```

Sunteţi acum gata să începeţi o conversaţie. Dar înainte de a face asta ar trebui să ştiţi câteva lucruri.

-Tot ce tipăriţi prin calea #4 este blocat: adică nu este imediat pus pe reţea, ci se aşteaptă până când se acumuleaza o anumită cantitate de date. Aşa că este necesară închiderea canalului prin CLOSE, imediat ce aţi terminat de tipărit. Astfel se transmite zona tampon chiar dacă nu este plină. (Zona tampon are lungimea 255 de octeţi sau caractere).

- Tot ce tipăriţi este marcat ca sosind în mod specific de la staţia la care lucraţi, astfel încât dacă staţia 2 este în aşteptare pentru un mesaj de la altă staţie, mesajul dumneavoastră va fi ignorat. Dacă mesajul emis este ignorat, ecranul nu va afişa mesajul OK, şi marginea ecranului va rămâne neagră până când mesajul este emis şi se primeşte confirmare pozitivă de la staţia 2.

- In timp ce instrucţiunea INPUT poate fi folosită pentru a aştepta ca să fie transmis ceva, funcţia INKEY$ poate fi folosită pentru a citi reţeaua. Se va întoarce cu primul octet din orice a fost transmis sau din orice aşteaptă să fie transmis. Altfel se întoarce cu şirul vid. Aceasta se numeşte interogare (pooling).

Programul de mai jos va tipări orice este transmis către staţia 1:

```
10 OPEN #8;"n";1
20 PRINT INKEY$#8;
30 GO TO 20
```

(Pentru mai multe detalii despre INKEY$ vezi secţiunea 3.15)

5.4. EMISIE GENERALĂ

Există un număr de staţie special, al cărui specificator este "n";0. Atunci când se aşteapta date de la staţia 0, veţi recepţiona orice mesaj care este emis către staţia 0. Iar atunci când scrieţi, mesajul emis către staţia 0 va fi recepţionat de oricine citeşte date de la un canal cu specificatorul "n";0.

Aceasta ar fi foarte util, de exemplu, într-o şcoala dacă fiecare elev ar avea un calculator, dar numai profesorul ar avea un minidrive.

Să presupunem că profesorul doreşte să emită un program. Mai întâi toţi elevii ar trebui să introducă:

```
LOAD *"n";0
```

Această comandă va face ca toate calculatoarele elevilor să intre în aşteptare pentru recepţia programului. Profesorul ar trebui să salveze programul în reţea introducând:

```
SAVE *"n";0
```

Emisiile generale, spre deosebire de mesajele private, încep imediat fără să aştepte ca alte calculatoare să fie gata să le recepţioneze.

De asemenea, la emisia generală, calculatorul nu vă poate informa dacă mesajul emis a fost recepţionat de către cineva.

Funcţia INKEY$ nu poate fi folosită pentru a interoga un canal de recepţie generală. La fel ca INPUT, ea va aştepta pur şi simplu să fie emis ceva.

5.5. JOCUL DE REŢEA

Pe discul de demonstraţie livrat odată cu sistemul există o copie a acestui joc. Numele de fişier este "net game". Programul este un bun exemplu pentru utilizarea reţelei. Părţi din el pot fi utile şi în programe scrise de dumneavoastră.

Jocul

Pentru a juca acest joc, cei doi parteneri trebuie să se gândească fiecare la un număr între 1 şi 100. Câştigătorul jocului este cel care ghiceşte primul numărul adversarului. La fiecare tentativă, calculatorul vă va spune cât de aproape sunteţi.

Programul

Subrutina de la linia 500 decide cine este utilizatorul 1 şi cine este utilizatorul 2. Asta este necesar deoarece atunci când se transmit tentativele, unul dintre voi foloseste subrutina de la linia 1100, iar celalalt subrutina de la linia 1200, şi astfel utilizatorul 1 trimite primul, iar utilizatorul 2 primeşte primul.

Programul decide cine este utilizatorul 1 trimiţând către celălalt calculator mesajul "1", şi apoi intrând în ascultare pe reţea. Dacă primeşte un "1", asta înseamnă că celălalt HC-2000 a pornit programul mai târziu. Primul HC-2000 trimite de aceea un "2" către calculatorul adversarului, şi se face singur utilizatorul 1. Dacă, pe de altă parte, programul primeşte înapoi un "2", asta înseamnă că celălalt program era deja pornit şi în aşteptare atunci când programul local a trimis "1". Programul local se face singur utilizatorul 2.

Dacă cele două programe pornesc în acelaşi timp, cele două mesaje "1" se vor ciocni pe reţea, programele se vor bloca amândouă în aşteptare, şi este necesară întreruperea unuia dintre programe cu BREAK şi restartarea.

Programul principal schimbă numele utilizatorilor, citeşte numărul secret (care

nu este trimis adversarului) și apoi compară tentativele. Mai întâi se transmite tentativa și apoi se afișează răspunsul.

Liniile de la 190 încolo detectează o victorie, o afișează corespunzător și apoi oferă un alt joc.

```
10 GO SUB 500
20 PRINT;: BORDER 1: PAPER 1: INK 7: CLS
30 PRINT " Joc de ghicit numere""Introduceţi mai intii un
   numar secret, apoi ghiciti-l pe al adversarului"
40 INPUT "Cum va numiti?";a$
50 PRINT ""Salut ";a$
60 GO SUB 1000 + 100*user
70 PRINT "Jucati cu ";b$
75 PRINT 'a$,b$
80 INPUT "Ghinditi-va la un numar (1 la 100)";a
90 IF a OR a100 OR aINT a THEN GO TO 80
130 INPUT "Ce numar incercati?";b
140 LET a$ = STR$ b: GO SUB 1000 + 100*user
150 LET c = ABS (a-VAL b$)
160 IF c = 0 THEN LET a$ = "Asta este" : GO TO 170
161 IF c THEN LET a$ = "Arde": GO TO 170
162 IF c THEN LET a$ = "Fierbinte": GO TO 170
163 IF c  THEN LET a$ = "Foarte cald": GO TO 170
164 IF c( THEN LET a$ = "Cald": GO TO 170
165 IF c« THEN LET a$ = "Rece": GO TO 170
166 LET a$ = "Gheata"
170 GO SUB 1000*100
180 PRINT b$,a$
190 IF c = 0 OR b$ = "Asta este" THEN GO TO 210
200 GO TO 130
210 IF b$ = "Asta este" THEN PRINT FLASH 1;"Victorie":
    FOR n = 0 TO 7: BORDER n: BEEP .1,n: BEEP .1,n + 16:NEXT n:
    GO TO 230
220 PRINT "Infringere": FOR n = 7 TO 0 STEP -1: BORDER n:
    BEEP .2,n: NEXT n
230 BORDER 1: INPUT "Alt joc? (d/n)";a$
240 IF a$ = "d" THEN RUN 20
250 STOP
500 OPEN #4;"n";0
510 PRINT #4;"1"
520 CLOSE #4
530 OPEN #4;"n";0
540 INPUT #4;a$
545 CLOSE #4
550 IF a$ = "1" THEN OPEN #4;"n";0: PAUSE 5: PRINT #4;"2":
    LET user = 1
560 if a$ = "2" THEN LET user = 2
```



```
570 CLOSE #4
580 FORMAT "n";user: RETURN
1100 OPEN #4;"n";3-user
1110 PRINT #4;a$
1120 CLOSE #4
1130 OPEN #4;"n";3-user
1140 INPUT #4;b$
1150 CLOSE #4
1160 RETURN
1200 OPEN #4;"n";3-user
1210 INPUT #4;b$
1220 CLOSE #4
1230 OPEN #4;"n";3-user
1240 PRINT #4;a$
1250 CLOSE #4
1260 RETURN
```

Capitolul 6. UTILIZAREA INTERFEŢEI SERIALE

6.1. CONECTAREA PERIFERICELOR LA INTERFAŢA SERIALĂ

Precum ştiţi, setul de caractere al lui HC-2000 conţine atât simboluri simple (litere, cifre, etc.) cât şi cuvinte cheie (instrucţiuni, nume de funcţii, etc.). Toate aceste caractere pot fi emise şi recepţionate prin interfaţa serială către/de la orice dispozitiv compatibil; de exemplu o imprimantă, un modem sau o altă interfaţă serială conectată la un tip diferit de calculator.

Pentru a conecta oricare din aceste periferice la interfaţa serială, trebuie să folosiţi un cablu cu un conector cu 9 pini la capătul dinspre HC-2000 şi un conector corespunzător dispozitivului la care vă cuplaţi la celalalt capăt. (Pentru detalii de interconectare vezi Canalul de serială.)

Apoi, înainte de a folosi interfaţa serială, va trebui să stabiliţi modul de lucru al perifericului:

- modul 'auto line feed' trebuie dezactivat. (HC-2000 va emite secvenţa 'retur car' (RETURN) şi 'avans rând' (LF) pe un canal "t", dar numai 'retur car' (RETURN) pe un canal "b". Aceste canale "t" şi "b" sunt explicate mai jos.)
- paritatea trebuie dezactivată.
- numărul de biţi trebuie stabilit la 8 (opt).
- numărul de biţi de stop trebuie stabilit la 1 (unu).
- viteza de emisie/recepţie (adică numărul de biţi pe secundă). HC-2000 poate comunica la oricare din vitezele standard, adică: 50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200.

Este bine să folosiţi cea mai mare viteză pe care o permite perifericul la care vă cuplaţi. (Veţi vedea mai jos cum puteţi face HC-2000 să folosească aceeaşi viteză.)

În astfel de momente, un manual de instalare pentru perifericul la care vă cuplaţi este foarte util.

6.2. CANALELE "t" şi "b"

Interfaţa serială are două tipuri de canale: canalul "t" şi canalul "b".

Canalul "t"

Canalul "t" (de la text) este folosit de obicei pentru a trimite listinguri. Canalul "t" are următorul efect asupra setului de caractere:

cod caracter

0-31: (caracterele de control) nu sunt emise, cu excepţia lui 13 (retur car) care este trimis 13 urmat de 10 (retur car şi avans rând).

32-126: (caracterele tipăribile) sunt trimise atare.

127-164: (caracterele grafice) nu sunt emise. Ele sunt înlocuite de caracterul ? (codul 63).

Pentru INPUT şi INKEY$ canalul "t" transferă numai caractere pe 7 biţi, aşa

încât forţează la 0 bitul 7.

Pentru a folosi canalul "t", trebuie mai întâi să stabiliţi viteza de lucru. Aşa că introduceţi:

```
10 FORMAT "t";viteza
              |
              +---- introduceţi aici viteza de lucru pe care aţi
                    stabilit-o şi la periferic.
```

Acum, pentru a deschide o cale către canalul "t", introduceţi:

```
20 OPEN #3,"t"
30 LLIST
```

Marginea ecranului va clipi şi listingul va fi trimis către periferic. (Remarcaţi că LLIST este este o prescurtare de la LIST #3.) Introduceţi acum:

```
LPRINT "Acesta este un mesaj."
```

Şi acest mesaj va fi trimis către dispozitiv.

Dacă HC-2000 este cuplat cu un terminal sau un calculator care poate trimite caractere, atunci puteţi citi date de la terminal sau calculator. Introduceţi:

```
10 FORMAT "t";viteza
20 OPEN #4,"t"
30 PRINT INKEY$#4;
40 GO TO 30
```

Acum, orice caracter primit de la terminal sau calculator va fi afişat pe ecran.

Canalul "b"

Canalul "b" (de la binar) trimite toţi cei 8 biţi ai codurilor folosite de HC-2000, şi vă permite să trimiteţi coduri de control către imprimante etc.

Şi la INPUT şi INKEY$ canalul "b" întoarce caractere pe 8 biţi.

SAVE şi LOAD funcţionează numai cu canalul "b".

Dacă aţi conectat două HC-uri prin interfaţa serială sau doriţi să vă memoraţi programele pe un alt tip de calculator care are de asemenea o interfaţa serială, veţi dori să salvaţi şi să încărcaţi programe prin interfaţa serială. Pentru aceasta introduceţi:

```
FORMAT "b",viteza
           |
           +---- introduceţi aici viteza pe care aţi
                 stabilit-o la periferic
```

Acum puteţi încerca, de exemplu:

```
10 REM cifre
20 FOR n = 1 TO 10
30 PRINT n,n*RND
40 NEXT n
```

urmat de:

```
SAVE *"b"
```

La celălalt capăt al legăturii cineva trebuie să introducă:

```
LOAD *"b"
```

Extensiile uzuale sunt de asemenea posibile:

```
SAVE *"b";SCREEN$
```

şi:

```
SAVE *"b";LINE numar
```

6.3. CUM SE TRIMIT CODURI DE CONTROL

Multe imprimante primesc secvenţe de control pentru operaţii de genul tipărire cu lăţime dublă. Pentru a trimite caracterele de control trebuie să folosiţi canalul "b". Atenţie însă, prin canalul "b" returul de car (RETURN) nu este urmat automat de avans rând (LF). De aceea este preferabil să aveţi două canale deschise, unul "b" şi altul "t": veţi folosi canalul "b" pentru a trimite secvenţele de control şi canalul "t" pentru texte. Să presupunem secvenţa de control pentru imprimare cu lăţime dublă este 14. Introduceţi:

```
10 OPEN #4;"b"
20 PRINT #4;"Latime normala ";
30 PRINT #4;CHR$ 14;"Latime dubla"
40 CLOSE #4
```

(Dacă exemplul nu funcţionează, căutaţi în manualul imprimantei codul pentru lăţime dublă.)

Încercaţi şi exemplul de mai jos:

```
10 OPEN #5;"b"
20 OPEN #6;"t"
30 PRINT #5; CHR$ 14;
40 LIST #6
50 CLOSE #5: CLOSE #6
```

Acest exemplu ar trebui să producă un listing pe lăţime dublă.

6.4. MODIFICĂRI RUTINA IMPRIMANTĂ

Controlul lăţime imprimare. În mod implicit înainte de tipărirea coloanei 81 a unei linii se emite secvenţa care forţează o linie nouă la imprimantă şi se resetează controlul intern de coloane. În acest fel lăţimea LIST-ingului program este limitată la 80 de coloane. De asemenea PRINT-urile foarte lungi sunt continuate pe linia următoare. Octetul 23729 din variabilele extinse conţine lăţimea imprimantei în coloane (implicit 80). Înainte de a modifica prin POKE această valoare trebuie să vă asiguraţi că variabilele extinse sunt inserate în sistem şi că aveţi REV.2. După instrucţiunile:

```
CLOSE #0: LET V = PEEK 23729
```

Variabila v va conţine:

- 0 pentru REV.1
- 80 pentru REV.2

După verificarea versiunii puteţi de exemplu modifica lăţimea imprimantei cu instrucţiunea: POKE 23729,64 la 64 de coloane.

Extinderea setului de caractere recunoscute cu operatorii TAB şi AT. Operatorul asigură tabularea datelor din 8 în 8 coloane. Operatorul TAB exp aduce poziţia de imprimare în coloana exp. Operatorul AT lin,col aduce poziţia de imprimare în coloana col. lin este ignorat. Numerotarea coloanelor pe imprimantă începe cu 0. Operatorii INK, PAPER, INVERSE, FLASH sunt ignoraţi.

Corecţii erori.

La închiderea unui stream de tip 't' se emite iar la închiderea unui stream de tip 'b' nu se emite nimic. Interfaţa veche emite un indiferent de tipul canalului 'b' sau't'.

La revizia 1 dacă operaţia de închidere a unui stream era întreruptă cu BREAK de la tastatură, în memorie rămâne o zonă cu octeţi nefolositori. La rev. 2 pana a fost eliminată.

Hook code nou

A fost inserată o nouă funcţie hook code: crează canal ad-hoc.

- Hook code +3C
- Intrarea 23769 tip canal:b/t
- Ieşirea reg de adresă canal.

Canalul fiind ad-hoc el este distrus în mod automat la revenirea in mod comandă.

6.5. INSTRUCȚIUNEA MOVE

Până acum am făcut transferuri de date de la un program către un canal sau invers. Instrucțiunea MOVE vă permite să mutați date de la un canal la altul. De exemplu, pentru a muta date de la claviatură la ecran, introduceți:

```
10 MOVE #1 TO #2
```

apoi:

```
RUN
```

Orice veți introduce de la claviatură va apare pe ecran. Dar veți descoperi că apăsarea lui BREAK nu face decât să tipărească un spațiu pe ecran. Pentru a ieși din această capcană, apăsați RETURN până ce ajungeți la ultima linie din ecran. Apoi, răspundeți cu BREAK la întrebarea scroll? (Ar trebui ca pe viitor să evitați să mutați date de la claviatură la orice altă cale pentru că s-ar putea să nu mai reușiți să ieșiți din instrucțiunea MOVE.)

Instrucțiunea MOVE se mai poate utiliza și pentru a examina fișierele memorate pe floppy. De exemplu, dacă mai aveți pe floppy fișierul "Numere", îi puteți examina conținutul cu instrucțiunea:

```
10 MOVE "d";1;"Numere" TO #2
```

(Remarcați că nu trebuie să deschideți/închideți (OPEN/CLOSE) fișierul, MOVE face singur lucrul acesta.)

De asemenea, pentru a face o copie a programului "Numere" introduceți:

```
10 MOVE "d";1;"Numere" TO "d";1;"Numere 2"
```

În acest caz, MOVE deschide o cale pentru a citi din fișierul existent ("Numere") și o altă cale pentru a scrie în fișierul nou creat ("Numere 2"). Apoi citește datele din "Numere" și le scrie în "Numere 2". Apoi închide ambele căi.

MOVE va funcționa atât cu numere de cale (de ex. #4), cât și cu specificatoare de canale (de ex. "d";1;"Numere"). Căile standard #1, #2 și #3 nu pot fi însă specificate cu numele consacrate K, S și P.

Puteți face o copie de siguranță a fișierului "Numere" pe alt disc folosind:

```
10 MOVE "d";1;"Numere" TO "d";2;"Numere 2"
```

Instrucțiunea MOVE poate fi folosită și pentru a trimite fișiere către imprimantă. Dacă aveți o imprimantă legată la interfața serială, introduceți:

```
10 FORMAT "t",viteza
20 OPEN #4,"t"
30 MOVE "d";1;"Numere" TO #4
```


6.6. PROGRAMUL PRINTER SERVER

Programul permite unui HC-2000 cuplat la o rețea să controleze o imprimantă serială. Imprimanta poate fi folosită de toate calculatoarele cuplate la rețea. Acest program este util dacă, de exemplu, un grup de utilizatori de HC-2000 posedă o singură imprimantă serială pe care vor să o împartă. Se arată totodată o utilizare mai deosebită pentru instrucțiunea MOVE.

Calculatorul folosit ca Printer Server trebuie să fie întotdeauna stația 64, și trebuie întotdeauna să facă legătura cu stația 62 (care este o stație specială de stabilire de contact). Astfel, stația emițătoare folosește temporar stația 62, și trimite numărul său real de stație, de la care va muta apoi un fișier spre canalul "t". Pentru a stabili un program Printer Server introduceți:

```
10 FORMAT "n";64
20 OPEN #4;"n";62: INPUT #4;a$: CLOSE #4
30 MOVE "n";CODE a$ TO "t"
40 OPEN #4;"b": PRINT #4;CHR$ 12: CLOSE #4: RUN
```

(Linia 40 trimite un avans de pagină.)

Programul de mai jos este cel folosit de emițător. Mai întâi, emițătorul stabilește temporar stația 62. Apoi se emite numărul real de stație al emițătorului. Apoi stația emțătorului revine la numărul său real. În final, linia 60 trimite datele care trebuiesc imprimate (în acest caz listingul).

```
10 LET statia = numar
                 |
                 +------ introduceți aici numărul de
                         stație al HC-ului local

20 FORMAT "n";62
30 OPEN #4;"n";64: PRINT #4;CHR$ statia: CLOSE #4
40 FORMAT "n";statia
50 OPEN #4;"n";64
60 LIST #4
70 CLOSE #4
```

Capitolul 7. BASIC-ul EXTINS

7.1. INSTRUCȚIUNILE CLEAR # ȘI CLS

Se recomandă folosirea instrucțiunilor CLS # și CLEAR # în prima linie a oricărui program.

Instrucțiunea CLEAR #

Așa cum instrucțiunea CLEAR șterge toate variabilele definite (operație care se execută în mod automat și la RUN), instrucțiunea CLEAR # șterge toate canalele și căile definite prin program, efectuând următoarele operații:

- decuplează toate căile de la canalele deschise de către utilizator
- eliberează spațiul de memorie ocupat de aceste canale. (Zona CHANS va conține numai canalele predefinite "k", "s" și "p".)
- cuplează căile #0, #1, #2 și #3 la canalele standard.- trece toate discurile în starea R/W (vezi mai jos).

Nu trebuie să se confunde efectul instrucțiunii CLEAR # cu efectul închiderii prin CLOSE # a tuturor căilor. Spre deosebire de CLOSE #, instrucțiunea CLEAR # abandonează pur și simplu datele care se găsesc în canale. Dacă, spre exemplu, se șterge prin CLEAR # un canal de disc prin care s-a scris într-un fișier, datele din ultimul buffer vor fi pierdute, și mai grav, nici datele care au fost deja scrise pe disc nu vor fi accesibile pentru citire.

Mesajul de eroare "Disk 'R/O'" și CLEAR #

Pentru a proteja datele înscrise pe floppy, HC-2000 utilizează o metodă de a preveni erorile datorate schimbărilor incorecte de floppy.

Dacă interfața detectează o schimbare de suport într-unul din minidrive-urile cu care a lucrat de la ultimul NEW sau CLEAR #, ea trece în mod automat discul respectiv în modul 'R/O' (numai citire).
Dacă după o astfel de schimbare se încearcă o operație care necesită scrierea de date pe acel disc, se va obține mesajul de eroare "Disk 'R/O'".
Pentru a corecta această situație, trebuie executată instrucțiunea CLEAR # înainte de orice schimbare de suport într-unul din Minidrive-uri.

Instrucțiunea CLS #

Efectele acestei instrucțiuni sunt similare cu execuția comenzilor:

```
PRINT;: BORDER 7: PAPER 7: INK 0: CLS
```


7.2. VARIABILE DE SISTEM

Pe lângă variabilele de sistem tabelate în capitolul 3, interfețele utilizează următoarele variabile:

Tip	Adresa	Nume	Conținut
X1	23734	FLAGS3	Biți de control interfețe
X2	23735	VECTOR	Adresa folosită pentru a extinde interpretorul BASIC
X10	23737	SBRT	Rutina de paginare a ROM-urilor
2	23747	BAUD	Număr pe 16 biți care determină rata de transfer pe linia serială calculată astfel: BAUD = (3500000/(26*baud rate))-2 O puteți folosi pentru a stabili viteze nestandard de comunicație serială.
1	23749	NTSTAT	Numărul stației locale pe rețea
1	23749	NTSTAT	Numărul stației locale pe rețea
1	23750	IOBORD	Biții 2..0 conțin culoarea marginii ecranului în timpul I/E prin interfețe. Puteți pune orice culoare doriți cu instrucțiunea POKE.
N2	23751	SER_FL	Spațiu de lucru de 2 octeți pentru interfața serială.
N2	23753	SECTOR	2 octeți nefolosiți
N2	23755	CHADDT	Salvare pentru indicator caracter curent
1	23757	NTRESP	Locație folosită pentru răspuns în rețea
1	23758	NTDEST	Început bloc de control în rețea. Conține numărul stației destinație 0-64.
1	23759	NTSRCE	Numărul stației sursă
X2	23760	NTNUMB	Numărul blocului 0-65535
N1	23762	NTTYPE	Tip bloc 0-normal 1-ultimul (EOF)
X1	23763	NTLEN	Lungime bloc de date 0-255
N1	23764	NTDCS	Suma de control pentru blocul de date
N1	23765	NTHCS	Suma de control pentru blocul de control
N2	23766	D_STR1	Începutul primentru blocul de date
N1	23765	NTHCS	Suma de control pentru blocul de control
N2	23766	D_STR1	Inceputul primmar cale 0-15
N1	23769	L_STR1	Tip dispozitiv "D", "N", "T" sau "B"
N2	23770	N_STR1	Lungime nume fișier
N2	23772	F_STR1	Adresa nume fișier
N8	23774	D_STR2	Al doilea specificator de 8 octeți folosit de MOVE și LOAD.
N1	23782	HD_00	Început zona de lucru pentru SAVE, LOAD, VERIFY și MERGE: cod tip de date 0 = prog, 1 = numere, 2 = sir, 3 = cod
N2	23783	HD_0B	Lungime bloc de date 0-65535
N2	23785	HD_0D	Adresa în memorie a blocului 0-65535
N2	23787	HD_0F	Lungime program fără variabile
N2	23789	HD_11	Numărul liniei de autostart
1	23791	COPIES	1 octet nefolosit

	23792		Începutul zonei CHANS
	23813		Începutul programului BASIC dar fără canale utilizator

OBSERVAŢII

1. Inserarea variabilelor de sistem se efectuează în mod automat la prima apariţie a unei erori, a unei comenzi specifice interfeţei 1 sau în cazul mesajului OK. Această inserare poate genera mesajul Out of memory dacă cei 58 de octeţi necesari nu sunt disponibili.

2. Deschiderea unei căi sau a unui canal de disc sau reţea necesită o anumită cantitate de memorie. Un canal de disc are 306 octeţi, iar un canal de reţea are 276. Aceste canale vor fi create fie prin OPEN # sau prin MOVE. Dacă RAMTOP este prea jos, aceste comenzi pot genera mesajul de eroare Out of memory.

3. Un alt efect al introducerii variabilelor de sistem sau al creerii canalelor este mutarea programelor în cod masina aflate în instrucţiuni REM. Puneţi întotdeauna aceste programe după RAMTOP.

7.3. CANALUL DE DISC

La fiecare deschidere a unui fişier prin una din instrucţiunile OPEN# sau MOVE, în zona denumită CHANS în manualul de BASIC se crează o zonă de memorie denumită canal. De obicei un canal este adresat în limbaj maşina de registrul IX. Canalul are o lungime de 306 octeţi şi conţine un bufer de 256 de octeţi.

Conţinutul canalului este următorul:

0		Adresa 8
2		Adresa 8
4		'D' sau 'D'+80H pentru un canal ad-hoc
5		Adresa rutinei de ieşire din ROM-ul din interfeţe
7		Adresa rutinei de intrare din ROM-ul din interfeţe
9		Lungime canal, adică 306
11	CHFLAG	0 = citire, 1 = scriere posibilă din/în acest canal
12	CHDRIVE	număr drive folosit de canal: 0 = curent, 1 = 1, 2 = 2
13	CHNAME	Numele fişierului completat cu spaţii până la 11 caractere. octet 9 bit 7 = r/o, octet 10 bit7 = sys
24		20 de octeţi folosiţi de sistemul de gestiune caractere
47	CHRR2	Indicator depăsire capacitate fişier în acces aleator
48	CHBYTE	Indicator caracter curent în buferul de date
50	CHDATA	256 octeţi pentru bufer

Deschiderea unui canal de disc nu crează o hartă de ocupare în memoria BASIC. Hărţile de ocupare disc există în permanenţă în memoria RAM instalată pe HC-2000 memorie comutată împreună cu ROM-ul din interfeţe.

7.4. CANALUL DE REŢEA

La deschiderea unei căi către reţea se creează o zonă de memorie denumită canal în spaţiul indicat de variabila de sistem CHANS. Această zonă este adresată în limbaj maşina de registrul IX. Canalul are o lungime de 276 octeţi şi conţine un buffer de 255 de octeţi.

Conţinutul canalului este următorul:

0		Adresa 8
2		Adresa 8
4		'N' pentru OPEN # sau 'N' + 80h pentru MOVE
5		Adresa rutinei de ieşire din ROM-ul din interfeţe
7		Adresa rutinei de intrare din ROM-ul din interfeţe
9		Lungime canal adică 276
11	NCIRIS	Numărul staţiei partenere în comunicaţie
12	NCSELF	Numărul staţiei locale la deschiderea canalului
13	NCNUMB	Numărul blocului 0-65535
14	NCTYPE	Tipul pachetului de date... 0 = normal,1 = ultimul
15	NCOBL	Numărul de octeţi în blocul de date
17	NCDCS	Suma de control pentru blocul de date
18	NCHCS	Suma de control pentru blocul de octeţi în blocul de date
19	NCDCS	Suma de control pentru blocul de date
20	NCHCS	Suma de control pentru blocul de ti utili din bufer
21	NCB	255 octeţi pentru buferul de date

7.5. CANALUL DE SERIALĂ

La deschiderea unei căi către interfaţa serială se crează o zonă de memorie denumită canal în spaţiul indicat de variabila de sistem CHANS. Această zonă este adresată în limbaj maşină de registrul IX. Canalul are o lungime minimă de 11 octeţi.

Conţinutul canalului este următorul:

0	Adresa 8
2	Adresa 8
4	'B' sau 'T'
5	Adresa rutinei de ieşire din ROM-ul din interfeţe
7	Adresa rutinei de intrare din ROM-ul din interfeţe
9	Lungime canal adică 11

7.6. CONEXIUNI INTERFAŢA SERIALĂ ŞI REŢEA

Conectorul de interfaţa serială este folosit partajat cu conexiunile pentru reţea precum urmează:

1. RxData (ieşire)
2. DTR intrare, trebuie să fie la nivel ridicat pentru 'gata'
3. CTS ieşire, este la nivel ridicat dacă este 'gata'
4. TxData (intrare)
5. NET1 conexiune reţea
6. Masa serială
7, 8. Masa serială
9. Masa comună reţea

Pentru conectare cu o interfaţa standard CCITT V24, în capătul celălalt al cablului trebuie folosit un conector cu 25 de pini cablat în felul următor:

2. TxData
3. RxData
5. CTS
6. +12v (DSR)
7. Masa
20. DTR

Ambele conexiuni pentru reţea sunt fire "calde" putând fi folosite împreună cu un fir de masă (de la pinul 8) pentru a conecta calculatorul la reţea.

7.7. MESAJE DE EROARE

Instrucţiunile implementate de interfeţe genereaza mesaje de eroare diferite de mesajele de eroare generate de ROM-ul din placa de bază. Aceste mesaje vor fi urmate de numărul liniei şi numărul comenzii din linia care a generat eroarea.

Aceste noi mesaje de eroare sunt listate mai jos în ordine alfabetică:

CODE error
Aţi încercat să încărcaţi (LOAD) un bloc de cod a cărui lungime este mai mare decât lungimea specificată de instrucţiunea LOAD.

Disk error
În timpul execuţiei unei operaţii de intrare/iesire pe disc a apărut o eroare care nu a putut fi reparată prin reîncercări.

Disk full
Aţi încercat să scrieţi date într-un disc care nu avea suficient spaţiu liber. Reîncercaţi programul cu un alt disc, sau eliberaţi spaţiu pe discul curent stergând fişierele de care nu mai aveţi nevoie.

Disk 'R/O'
Aţi încercat să efectuaţi o operaţie de scriere pe un suport schimbat, fără să comunicaţi calculatorului prin CLEAR # faptul că aţi terminat de lucrat cu vechiul suport. Introduceţi CLEAR # şi apoi repetaţi comanda.

Disk 'write' protected
Aţi încercat o operaţie de scriere pe un disc care are montată protecţia la scriere. Îndepărtaţi protecţia şi apoi reîncercaţi.

File not found
Aţi încercat o operaţie asupra unui fişier inexistent, sau aţi încercat o operaţie CAT pe un disc fără nici un fişier.

File 'R/O'
Aţi încercat să stergeţi sau să scrieţi un fişier care are atributul de protejat la scriere (octetul 9 din nume bitul 7=1). Deprotejaţi fişierul dacă sunteţi sigur că vreţi să-l modificaţi.

Invalid device expression
Aţi specificat un dispozitiv diferit de k, s, p, d, n, b sau t. Acelaşi mesaj se obţine dacă aţi folosit punct-virgula în loc de virgula pentru unul din specificatorii k, s sau p.

Invalid drive number
Aţi specificat un număr de Minidrive mai mare 2, sau aţi specificat numărul 0 (Minidrive-ul curent), înainte de a-l declara printr-un apel explicit.

Invalid name
Numele fişierului este fie un şir vid, fie are mai mult de unsprezece caractere.

Invalid station number
S-a specificat un număr de staţie în afără domeniului 0-64 (1-64 pentru instrucţiunea FORMAT).

Invalid stream number
Numărul de cale specificat este în afără domeniului 0-15.

MERGE error
Aţi încercat să comasaţi date sau cod. MERGE funcţionează numai cu programe.

Missing baud rate
Lipseşte rata de transfer în instrucţiunea FORMAT "b" sau "t".

Missing drive number
Lipseşte numărul minidrive-ului.

Missing name
Lipseşte numele fişierului.

Missing station number
Lipseşte numărul staţiei în reţea.

Program finished
Aţi încercat să executaţi o linie dincolo de ultima linie din program. Acest mesaj de eroare va apare dacă executaţi un GO TO urmat de un număr de linie mai mare decât ultima linie din program. Va apare de asemenea dacă introduceţi RUN fără a avea un program în memorie.

Reading a 'write' file
Incercaţi să citiţi date dintr-un fişier disc inexistent, sau dintr-un canal care a fost deja folosit pentru scriere.

Stream already open
Aţi încercat să deschideţi o cale care a mai fost folosită pentru un canal de tip nou (d, n, t sau b). Calea poate fi deschisă numai după ce a fost închisă.

Verification has failed
Există diferenţe între fişierul salvat şi programul, datele sau codul existente în memorie.

Writing to a 'read' file
Aţi încercat să scrieţi într-un fişier disc existent. Fişierul existent trebuie mai întâi şters, dacă nu este nevoie de el. Altfel trebuie utilizat un fişier nou.

Wrong file type
Aţi încercat să încărcaţi (LOAD) un fişier de date sau cod pe un program sau invers, un program pe un fişier de date sau cod.

7.8. INSTRUCŢIUNILE BASIC-ULUI EXTINS

Interfeţele extind BASIC-ul existent deja în HC-2000. Extensiile şi adăugările sunt rezumate mai jos.

Căile

Căile sunt specificate prin #n unde n este un număr în domeniul 0-15. Căile 0, 1, 2 şi 3 sunt de obicei folosite de BASIC. Caracterul # este parte din cuvântul cheie pentru instrucţiunile OPEN # şi CLOSE #.

Canalele

Există şapte tipuri de canale în BASIC-ul extins: claviatura (k), ecranul (s), imprimanta (p), interfaţa serială pentru texte (t), interfaţa serială binară (b), reţeaua (n) şi discul (d).

Fiecare canal este specificat prin litera lui care poate fi majusculă sau nu. Reţeaua şi discul au nevoie de informaţii suplimentare pentru a specifica complet canalul.

Un canal de reţea necesită un număr de staţie, aşa încât un specificator de reţea are forma "n";x unde x este numărul staţiei în domeniul 0-64.

Un canal de disc necesită un număr de minidrive şi un nume de fişier care trebuie să fie un şir cu 1 până la 11 caractere.

Minidrive curent

Primul minidrive la care se face acces dupa NEW sau CLEAR # va deveni ceea ce se numeşte minidrive-ul curent. Din acest moment înainte acest minidrive poate fi specificat şi cu numărul 0.

Dacă se încearcă folosirea specificatorului 0 înainte de a defini minidrive-ul curent, se va semnala eroarea Invalid drive number.

Caracterul ? în nume de fişier

Este bine să nu folosiţi caracterul ? în numele de fişiere, pentru că acest caracter are alt rol. El este folosit de sistemul de fişiere pe post de "Jolly Jocker" (wild card), putând să înlocuiască orice alt caracter, dar unul şi numai unul.

Dacă de exemplu trebuie să ştergeţi fişierele cu numele "nume0", "nume1", "nume2" şi nu aveţi alte fişiere cu nume de forma "numex" (unde x este orice caracter), puteţi folosi o singură comandă de forma ERASE "d",1,"nume?" care şterge toate cele trei fişiere.

Dacă într-un OPEN # specificaţi parţial un nume de fişier existent (folosind caracterul "?") va fi folosit primul fişier al cărui nume se potriveşte.

Instrucţiuni

CAT y — Listează toate numele fişierelor aflate pe floppy-ul din minidrive-ul y. Lista este prezentată în ordinea din catalogul discului şi din minidrive-ul y.

CAT #z;y — Trimite catalogul floppy-ului din minidrive-ul y către calea z.

CAT y;"cc...c" — Listează numele fişierelor care se potrivesc cu şirul de caractere "cc...c", care poate conţine "?" pentru specificări ambigue.

CAT #z;y;"cc...c" — Ca mai sus, dar trimite lista către calea z.

CLEAR # — Readuce sistemul de căi şi canale la starea de dupa NEW - există numai canalele standard k, s, p şi sunt deschise numai căile standard #0, #1, #2 şi #3. Eventualele date existente în canalele utilizator sunt ignorate, spaţiul de memorie fiind eliberat fără remuşcări.

CLOSE #cale Desface legătura dintre calea specificată şi orice canal. Dacă există date blocate prin scriere în bufferul canalului atunci acestea sunt fie transmise (pe reţea) sau înregistrate (pe floppy).

CLS # Readuce ecranul în starea de după NEW. BORDER alb, PAPER alb, INK negru, ecran şters.

ERASE "d";y;"nume" Şterge fişierele specificate de nume aflate pe discul din minidrive-ul y. Numele poate conţine caracterul "?" pentru specificaţii ambigue.

FORMAT "d";y Pregăteste un floppy din minidrive-ul y pentru a fi utilizat din BASIC.

FORMAT "n";x Stabileşte numărul staţiei pe reţea la x.

FORMAT "t";x Stabileşte viteza de comunicaţie pentru FORMAT "b";x interfaţa serială la x (x trebuie ales dintre vitezele standard de comunicaţie 50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200).

INKEY$#cale Întoarce un singur caracter sub forma unui şir dacă cel puţin unul este disponibil sau întoarce sirul vid "" dacă nu exista caracter disponibil din calea respectivă. Aceasta instrucţiune are sens doar dacă calea este legată la un canal de reţea sau de interfaţa serială.

INPUT# cale;var Citeşte variabila var din calea specificată. Calea trebuie să fi fost deschisă înainte către un canal de intrare. Este important să reţineti că orice element de PRINT care apare în instrucţiunea INPUT va fi scris către această cale. Aceasta este de obicei necesar numai atunci când se citesc date de la claviatură. Reţineti de asemenea că separatorul ";" scrie un caracter. Opţiunea LINE este disponibilă ca mai înainte.

LOAD *canal optiuni Încarcă programul, datele sau codul de la canalul specificat. Se pot folosi numai canalele "b", "n" sau "d".
Toate opţiunile existente pentru LOAD sunt disponibile şi la LOAD *.

MERGE *canal optiuni La fel ca LOAD, doar că nu şterge liniile de program sau variabilele decât pentru a face loc pentru unele noi cu acelaşi număr de linie sau nume.

MOVE sursa TO destinatie
Muta datele de la sursă către destinaţie.
Sursa şi destinaţia pot fi numere de cale sau canale. Comanda se termină numai la întâlnirea unui indicator de sfârşit de fisier în sursă: aceasta se poate întâmpla doar dacă sursa este un canal de reţea sau disc, sau altfel o cale legată la un astfel de canal.
Dacă sursa sau destinaţia sunt specificate drept canale, atunci acestea sunt deschise la început şi închise la terminarea transferului.

OPEN #cale,canal Leagă calea specificată la canalul specificat pentru a permite programului BASIC să citească sau să scrie din/în acel canal. Calea trebuie să fie închisă sau deschisă către unul din canalele k, s sau p.

PRINT #cale... Tipăreşte secvenţa de PRINT către calea specificată. Calea trebuie să fi fost deschisă în prealabil către un canal de ieşire.
Secvenţa de PRINT poate avea aceeaşi sintaxă ca mai înainte şi poate conţine alte elemente de tipul #cale.

SAVE *canal optiuni Salvează programul, datele sau codul către canalul specificat. Pot fi folosite numai canalele "b", "n" sau "d".
Toate opţiunile existente la SAVE sunt disponibile şi la SAVE *.

VERIFY *canal opt La fel ca LOAD (vezi mai sus) cu excepţia faptului că datele nu sunt încărcate în memorie, ci sunt doar comparate cu ceea ce exista deja acolo.

ICE FELIX
COMPUTER S.A.

78009 BUCHAREST sect 2
Str.G.Constantinescu no.2
Tel. : 688 38 00

Fax : 687 62 20, 312 87 50
Tel. : 688 22 95, 688 61 25
Tel. : 687 53 02, 688 23 60